University of Málaga

Doctoral Thesis

Automated Recommendation of Multi-Objective Optimization Algorithms Using a Knowledge-Based Approach

Author

José Francisco Aldana Martín

Tutor and Supervisor

Dr. Antonio Jesus Nebro Urbaneja

Supervisor

Dra. María del Mar Roldán García

UNIVERSIDAD DE MÁLAGA

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

DOCTORAL THESIS
COMPUTER TECHNOLOGIES

Automated Recommendation of Multi-Objective Optimization Algorithms Using a Knowledge-Based Approach

E.T.S.I. Informática
R.D. 99/2011

Author

José Francisco Aldana Martín
Khaos Research Group
ITIS Software
University of Málaga

Tutor and Supervisor

Dr. Antonio Jesus Nebro Urbaneja
ITIS Software
Department of
Computer Science and Programming Languages
University of Málaga

Supervisor

Dra. María del Mar Roldán García
ITIS Software
Department of
Computer Science and Programming Languages
University of Málaga

July, 2024

Automated Recommendation of Multi-Objective Optimization Algorithms Using a Knowledge-Based Approach

Front and back cover design by Serafín Cortés Ramírez.

UNIVERSIDAD
DE MÁLAGA

AUTOR: José Francisco Aldana Martín

https://orcid.org/0000-0002-4845-762X

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D. JOSÉ FRANCISCO ALDANA MARTÍN
Estudiante del programa de doctorado TECNOLOGÍAS INFORMÁTICAS de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: **"AUTOMATED RECOMMENDATION OF MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS USING A KNOWLEDGE-BASED APPROACH"**.

Realizada bajo la tutorización de ANTONIO JESÚS NEBRO URBANEJA, y dirección de ANTONIO JESÚS NEBRO URBANEJA Y MARÍA DEL MAR ROLDÁN GARCÍA.

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 2 de JUNIO de 2024

Fdo.: JOSÉ FRANCISCO ALDANA MARTÍN Doctorando/a	Fdo.: ANTONIO JESÚS NEBRO URBANEJA Tutor/a
Fdo.: ANTONIO JESÚS NEBRO URBANEJA Y MARÍA DEL MAR ROLDÁN GARCÍA Director/es de tesis	

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

DECLARACIÓN DE DIRECCIÓN Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D. ANTONIO JESÚS NEBRO URBANEJA y Dña MARÍA DEL MAR ROLDÁN GARCÍA, profesores doctores del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

DECLARAN QUE:

D. JOSÉ FRANCISCO ALDANA MARTÍN, estudiante del programa de doctorado TECNOLOGÍAS INFORMÁTICAS ha realizado bajo su dirección y tutorización la tesis presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: **"AUTOMATED RECOMMENDATION OF MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS USING A KNOWLEDGE-BASED APPROACH"**.

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo. Así mismo, las publicaciones en coautoría que avalan dicha tesis no forman parte de otra tesis doctoral en la Universidad de Málaga ni en ninguna otra universidad.

En Málaga, a 2 de JUNIO de 2024

Fdo.: ANTONIO JESÚS NEBRO URBANEJA Y MARÍA DEL MAR ROLDÁN GARCÍA
Director/es de tesis

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

DECLARACIÓN DE TUTORIZACIÓN Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D. ANTONIO JESÚS NEBRO URBANEJA, profesor doctor del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

DECLARA QUE:

D. JOSÉ FRANCISCO ALDANA MARTÍN, estudiante del programa de doctorado TECNOLOGÍAS INFORMÁTICAS ha realizado bajo su tutorización la tesis presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: **"AUTOMATED RECOMMENDATION OF MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS USING A KNOWLEDGE-BASED APPROACH"**.

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo. Así mismo, las publicaciones en coautoría que avalan dicha tesis no forman parte de otra tesis doctoral en la Universidad de Málaga ni en ninguna otra universidad.

En Málaga, a 2 de JUNIO de 2024

Fdo.: ANTONIO JESÚS NEBRO URBANEJA
Tutor de tesis

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

Acknowledgements

My dear people, my dear family Aldana and Montes, Martín and López, and my dear Vázquez-Pendón, and Burgueño-Romero and Benítez-Hidalgo, and Cardas-Ezeiza, Cortés-Ramírez, Nebro-Urbaneja, Roldán-García, Durillo-Barrionuevo and Del-Ser-Lorente. Today I finish my PhD thesis: I am a doctor today!

I hope you are all enjoying yourselves as much as I am. I shall not keep you long. I have called you all together for a purpose. Indeed, for three purposes!

First of all, to tell you that I am immensely fond of you all, and that the three years of this PhD is too short a time to live among such excellent and admirable hobbits.

I don't know half of you half as well as I should like; and I like less than half of you half as well as you deserve.

Secondly, to celebrate the end of my PhD. I should say: OUR PhD. For lately has also marked the end of some of my PhD colleagues.

It is also, if I may be allowed to refer to ancient history, the fifth anniversary of my arrival to Khaos Research; the research group were I have worked on this PhD thesis. The kinder palmier was very splendid, however, though I had a bad cold at the time, I remember, and could only say 'thag you very buch'. I now repeat it more correctly: Thank you very much for coming to my little party.

I wish to make an ANNOUNCEMENT.

I regret to announce that - though, as I said, one PhD is far too short a time to spend among you - this is the END. I am going. I am leaving NOW.

GOOD-BYE!

Adapted from Bilbo Baggins' birthday speech by J.R.R. Tolkien in The Lord of the Rings, The Fellowship of the Ring, "A Long-expected Party".

Table of Contents

List of Figures

List of Tables

List of Listings

Abstract

This PhD thesis, titled "Automated recommendation of multi-objective optimization algorithms using a knowledge-based approach", addresses the challenge of developing a tool to provide algorithmic recommendation to end users (experts in the problem domain but not experts in multi-objective algorithms) without the need of a resource-intensive process of auto-configuration. This challenge is faced with an approach based on previous knowledge about the problems.

A semantic model, *moody*, is designed to formally define knowledge in the field of multi-objective optimization with metaheuristics, with a focus on the relevant concepts required to characterize problems and the performance of algorithms.

moorphology is developed as a tool to provide landscape characteristics of the search and objective spaces of multi-objective problems. These landscape characteristics are a key factor for the computation of a similarity metric between multi-objective problems, which are a necessity to provide recommendations based on previous knowledge.

To generate in an efficient way the required knowledge to implement the recommendation engine, a meta-optimization approach is presented as the software tool *Evolver*. This tool allows the automatic configuration of metaheuristics by defining it as an optimization problem.

Large language models are evaluated for the task of helping domain experts in implementing their problems into an optimization framework for solving them. To solve this problem, a large language model is fine-tuned and embedded into a graphical tool, named *moostral*, to allow the end user to easily implement their optimization framework into the recommendation system described in this thesis.

To connect the previously mentioned elements, a recommendation engine, named *recommoonder*, is implemented to solve the challenge presented in this thesis.

Finally, several future work lines are identified, such as: extending the semantic model to support new metaheuristics, improving the similarity metric by using machine learning or adapting the proposed auto-configuration process for problems with unknown Pareto fronts. This thesis has a very practical focus, providing open source repositories for all the tools developed in it, allowing their use for further research in the defined lines.

Resumen en español

Este capítulo proporciona una visión general de los principales campos explorados en esta tesis doctoral, ofreciendo una visión general de la motivación que ha llevado a esta investigación, junto con sus principales metas y objetivos. A continuación, se destacan las principales contribuciones de esta tesis doctoral a los campos relacionados, además de esbozar la organización de los capítulos restantes de la tesis. Por último, se describen las conclusiones y los futuros trabajos derivados de esta tesis, resumiendo los principales resultados y destacando su relevancia en cada uno de los campos relacionados. Sobre los resultados y las limitaciones identificadas, la atención se centra a continuación en las posibles áreas de investigación futura.

Antecedentes

La optimización ha sido una parte integral de como los humanos hemos resuelto problemas durante siglos, incluso precediendo a las figuras históricas más antiguas conocidas. Los problemas de optimización, que implican encontrar la mejor solución entre todas las soluciones factibles, se han abordado de diferentes formas desde la antigüedad. Por ejemplo, en la antigua Grecia, Euclides sentó las bases para la optimización geométrica al definir la distancia euclídea, que representa la distancia mínima entre dos puntos en línea recta. Esta perspectiva de Euclides se puede enmarcar como un problema de optimización donde el objetivo es minimizar la distancia entre dos puntos. Al formularlo de esta manera, se pueden aplicar técnicas matemáticas para encontrar la solución óptima, que resulta ser la línea recta que conecta los dos puntos. Este concepto de optimización, arraigado en los principios fundamentales de la geometría, ha trascendido el tiempo y las disciplinas, convirtiéndose en un pilar en campos que van desde las matemáticas y la física hasta la ingeniería y la informática.

Hasta la década de los ochenta del siglo XX, las técnicas de optimización eran principalmente métodos matemáticos clásicos. La década de ochenta marcó un giro significativo en el campo con el surgimiento de algoritmos evolutivos [28, 33]. Estos algoritmos, inspirados en la evolución biológica, fueron particularmente buenos para resolver problemas de optimización complejos que anteriormente eran intratables. Los algoritmos evolutivos representan técnicas estocásticas que, si bien no garantizan la identificación del óptimo global para los problemas, a menudo producen soluciones cuasióptimas dentro de un período de tiempo razonable. Notablemente, estos algoritmos no exigen necesariamente información específica del dominio sobre el problema que se está optimizando para generar aproximaciones robustas a la solución óptima. Estas características contribuyen a su amplia adopción, convirtiéndolos en técnicas altamente utilizadas en diversos dominios.

Durante la década de los noventa, los algoritmos evolutivos se convirtieron en parte de una categoría más amplia conocida como las metaheurísticas. Estas técnicas de alto nivel operan mediante la manipulación de métodos de nivel inferior, normalmente heurísticas, para ofrecer algoritmos de optimización eficientes. Esta integración amplió el alcance de los enfoques de optimización, abarcando diversos algoritmos, como la Optimización por Enjambre de Partículas (PSO), la Optimización de Colonias de Hormigas (ACO), la Búsqueda Tabú (TS), y muchas otras [17]. Durante las últimas dos décadas, el enfoque de la investigación se ha intensificado en el estudio de problemas de optimización multiobjetivo y su resolución utilizando metaheurísticas. Estos problemas, frecuentemente en-

contrados en aplicaciones de diversos dominios del mundo real, se caracterizan por estar formulados en función de dos o más objetivos conflictivos que deben optimizarse simultáneamente. Esto implica que mejorar uno de ellos conduce a un empeoramiento en el resto [45].

El desafío de la optimización multiobjetivo radica en navegar por la naturaleza contradictoria de estos objetivos, donde la solución óptima no es un único resultado sino un conjunto de soluciones de compromiso, conocido como conjunto de Pareto. En este conjunto, ninguna solución supera a otra en todos los objetivos, y su correspondencia en el espacio objetivo es lo que se denomina frente de Pareto. Cuando se aplican para resolver problemas multiobjetivo, las metaheurísticas tienen como objetivo encontrar una aproximación precisa al frente de Pareto.

El complejo panorama de la optimización está, además, condicionado por el teorema de *No Free Lunch* [160], que sostiene que no hay un algoritmo universalmente óptimo para resolver todos los problemas de optimización dentro de una clase. Esto implica que, cuando se enfrenta a un problema de optimización específico, no se puede determinar de antemano el algoritmo óptimo para abordarlo. La naturaleza intrincada de los detalles específicos del problema exige un enfoque dinámico y adaptable para la selección de algoritmos. Al mismo tiempo, las metaheurísticas son extremadamente sensibles a los parámetros de configuración, dependiendo en gran medida del ajuste de parámetros para aumentar la efectividad de estas técnicas. Esto requiere una comprensión profunda del campo y, lo que es más importante, del algoritmo específico utilizado.

Los usuarios finales (biólogos, ingenieros, economistas, etc.) que necesitan optimizar problemas multiobjetivo no suelen ser expertos en metaheurísticas, y suelen utilizar la configuración predeterminada de un algoritmo popular sin ajustar sus parámetros a su problema del mundo real. Por lo tanto, un área de investigación actual y activa implica el desarrollo de herramientas que permitan a estos usuarios encontrar fácilmente un algoritmo (y configuración) capaz de resolver efectivamente sus problemas. En este sentido, se están investigando técnicas para la ajuste automático de parámetros de metaheurísticas [73, 122] e incluso para el diseño automático de algoritmos [13]. Estas técnicas tienen como objetivo adaptar automáticamente los parámetros de las metaheurísticas, optimizando su rendimiento para problemas específicos. Sin embargo, tienen la desventaja de tener unos requisitos de recursos computacionales muy elevados. Este área de investigación está en constante evolución, buscando cerrar la brecha entre la complejidad de los problemas del mundo real y la efectividad de las técnicas de optimización.

En este contexto, esta tesis doctoral propone un enfoque alternativo basado en el uso del conocimiento existente para crear una herramienta que brinde recomendaciones a los usuarios finales [161] sobre configuraciones superiores de algoritmos más allá de sus ajustes predeterminados. Sobre esta base, este trabajo aprovecha la eficacia establecida de las tecnologías de la Web Semántica, que han demostrado su eficacia en muchos otros campos. Estas tecnologías son fundamentales para integrar y representar el conocimiento del dominio, apoyar la estandarización de datos y facilitar la integración semántica desde múltiples fuentes [134]. Las ontologías son el método más ampliamente utilizado para describir el conocimiento, ya que permiten una definición formal y basada en la lógica de primer orden como un vocabulario común para compartir información en un dominio específico [145]. Para alimentar las recomendaciones, se utiliza una ontología para definir la estructura de un grafo de conocimiento donde se pueden almacenar los resultados de las herramientas de generación de configuraciones. Para facilitar aún más a los usuarios finales el uso de metaheurísticas, se estudian los Modelos Grandes de Lenguaje (siendo LLMs sus siglas en inglés) [171] como una tecnología para ayudar con la implementación de problemas de optimización multiobjetivo dentro de marcos de optimización.

Motivación

La principal hipótesis que sustenta esta tesis doctoral es que, dada la experiencia previa sobre la relación entre una configuración algorítmica específica y la calidad del resultado de dicho algoritmo para resolver un problema, y dada una métrica de similitud entre dos problemas, es posible proporcionar recomendaciones a usuarios no expertos para elegir una configuración algorítmica para resolver eficientemente un problema específico.

El uso de un enfoque basado en el conocimiento es clave en este trabajo. Las ontologías permiten la descripción inequívoca de entidades y las relaciones entre estas entidades en un dominio de aplicación. La definición de un modelo semántico basado en una ontología OWL 2 (*Web Ontology Language*) [58] para la configuración automática de parámetros de algoritmos de optimización facilita aprovechar el resto de tecnologías de la Web Semántica.

El uso de tecnologias de la Web Semantica posibilita no solo describir formalmente problemas, algoritmos, parámetros, configuraciones y métricas de calidad, sino también anotar los resultados de las herramientas de generación de configuraciones e integrar toda esta información fácilmente en un grafo de conocimiento. Además, las ontologías permiten definir restricciones lógicas en el dominio, lo que ayuda a verificar la validez de las configuraciones, facilita la con-

sulta del grafo de conocimiento y ofrece la posibilidad de aplicar razonamiento semántico sobre él [71].

Esta línea de investigación se enmarca dentro del contexto del proyecto "Aether-UMA: Una aproximación holística de smart data para el análisis de datos guiado por el contexto: explotación de semántica y contexto" (PID2020-112540RB-C41), financiado por el Ministerio de Ciencia e Innovación de España, ya que sus objetivos están fuertemente alineados con él.

Objetivos y fases

El objetivo principal de esta tesis es el diseño y desarrollo de una herramienta de recomendación para metaheurísticas para resolver problemas de optimización continuos multiobjetivo mediante un enfoque basado en el conocimiento utilizando las tecnologías de la Web Semántica. Este objetivo se divide en varios objetivos más pequeños de la siguiente manera:

Objetivo 1: Diseñar e implementar una ontología OWL 2 para representar el conocimiento en el campo de la optimización multiobjetivo, tanto para los algoritmos como para los problemas, que sirva como modelo semántico para generar un grafo de conocimiento.

- 1.1: Seleccionar un conjunto representativo de algoritmos de optimización multiobjetivo para modelar sus parámetros.
- 1.2: Selección de un conjunto de problemas multiobjetivo de referencia y las características a estudiar sobre ellos.
- 1.3: Diseñar e implementar la ontología.

Objetivo 2: Generación automática de configuraciones de los algoritmos y problemas elegidos previamente.

- 2.1: Evaluar un enfoque de meta-optimización para la generación y evaluación de configuraciones algorítmicas.
- 2.2: Diseñar e implementar la metodología de anotación para agregar configuraciones y características del problema al grafo de conocimiento.

Objetivo 3: Diseñar un conjunto de consultas SPARQL para buscar, para un problema dado, los algoritmos y configuraciones más prometedores para resolverlo.

- 3.1: Diseñar las consultas SPARQL para proporcionar recomendaciones a los usuarios.

3.2: Experimentación y validación.

Objetivo 4: Definir o seleccionar una métrica para evaluar la similitud entre varios problemas multiobjetivo, según sus características del *landscape* de los espacios de busqueda y objetivos.

4.1: Extracción de las características del *landscape* de problemas multiobjetivo.

4.2: Diseño de una métrica a partir de las características disponibles para definir la similitud entre varios problemas a partir de sus características del *landscape*.

4.3: Experimentación y validación de la métrica elegida.

Objetivo 5: Implementar herramientas auxiliares para facilitar la implementación de problemas de optimización multiobjetivo dentro de un marco compatible con el sistema de recomendación.

5.1: Evaluar el uso de LLM existentes para la implementación automática de problemas de optimización multiobjetivo a partir de una representación textual.

5.2: Generar un conjunto de datos de alta calidad de problemas sintéticos multiobjetivo para permitir el entrenamiento de un modelo.

5.3: Adaptar un LLM para resolver esta tarea.

5.4: Evaluar el modelo adaptado sobre un conjunto de problemas del mundo real.

Objetivo 6: Diseño e implementación de una herramienta de recomendación

6.1: Implementar la herramienta de recomendación.

6.2: Pruebas y validación de las recomendaciones proporcionadas.

Además, como objetivo no funcional se plantea proporcionar implementaciones de código abierto de todas las contribuciones relacionadas con esta tesis doctoral, siguiendo las mejores prácticas en desarrollo de software y proporcionando documentación de alta calidad para cada proyecto. Este enfoque facilita la utilización de estas implementaciones, tanto dentro como fuera de la comunidad de investigación.

Contribuciones

Las principales contribuciones de esta tesis doctoral están relacionadas con los objetivos descritos en la sección anterior de la siguiente manera:

- Un *framework* semántico guiado por una ontología para consolidar el conocimiento de la optimización multiobjetivo en forma de una ontología OWL 2 siguiendo los principios FAIR [157]. El Capítulo 3 presenta la ontología *moody* (Multi-Objective Optimization ontologY), la columna vertebral semántica del *framework* semántico que aborda los **Objetivos 1** y **3**.
- En el Capítulo 4, se realiza un estudio sobre el *landscape* del espacio de búsqueda de los problemas multiobjetivo seleccionados (**Objetivo 4**). Se ha seleccionado un conjunto de métricas que se utilizarán para caracterizar el *landscape* del espacio de variables y objetivos, e implementar *moorphology*, una biblioteca de software para calcular estas métricas en problemas implementados en el *framework* jMetal.
- El Capítulo 5 cubre el **Objetivo 2**, revisando el estado del arte en autoconfiguración para optimización multiobjetivo y proponiendo una herramienta destinada a configurar y diseñar automáticamente metaheurísticas para problemas de optimización multiobjetivo, llamada *Evolver*. La clave de *Evolver* es formular el proceso de autoconfiguracion como un problema de optimización multiobjetivo, lo que permite la aplicación de algoritmos de optimización multiobjetivo ya establecidos para encontrar configuraciones específicas para metaheurísticas.
- Una evaluación de los LLMs en el estado del arte para resolver la tarea descrita en el **Objetivo 5** se presenta en el Capítulo 6. Después de la evaluación, el capítulo se centra en el resto de los objetivos propuestos: se describe el desarrollo de una herramienta alimentada por un LLM para la implementación automática de problemas de optimización multiobjetivo, incluida la generación de un conjunto de datos de entrenamiento, el proceso de ajuste fino y la validación posterior del modelo.
- El **Objetivo 6** se aborda en el Capítulo 7, en el que se desarrolla *recommoonder*, una herramienta de recomendación que integra las demás contribuciones de esta tesis doctoral en un sistema de recomendación integrado para ofrecer a los usuarios mejores configuraciones de algoritmos que las que se usa por defecto.

Una descripción visual de estas contribuciones se puede ver en la Figura 1.

El desarrollo de esta tesis doctoral ha dado lugar a la publicación de dos

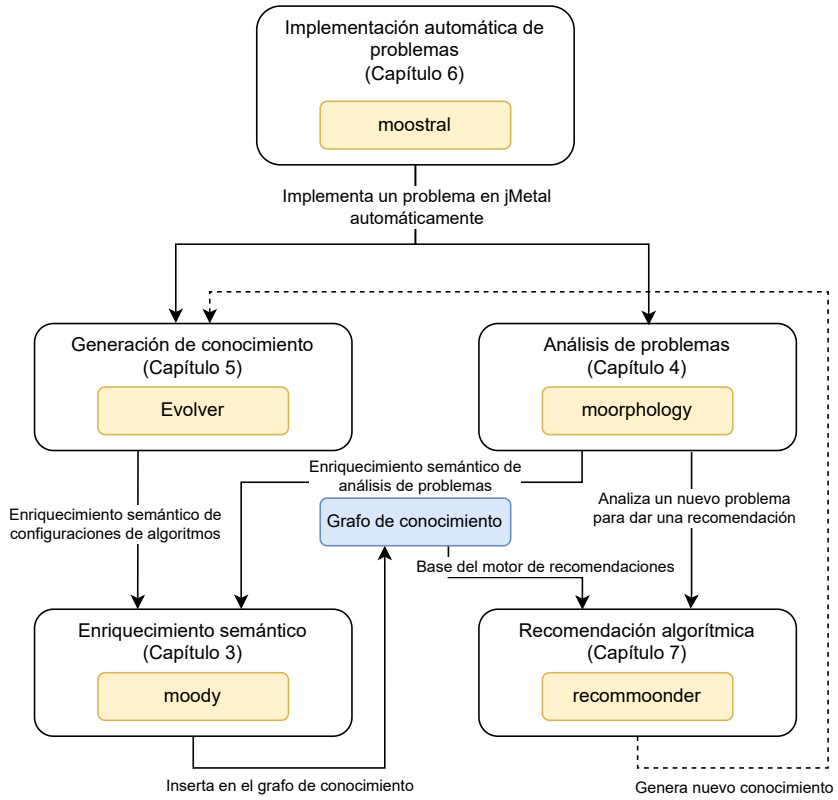


Figura 1: Descripción general de las contribuciones de esta tesis doctoral.

manuscritos en revistas de alto impacto, indexadas en el *Journal Citation Reports* (JCR), y otro artículo presentado en una conferencia internacionale:

- José F. Aldana-Martín, Juan J. Durillo, Antonio J. Nebro. **"Evolver: Meta-optimizing multi-objective metaheuristics"**. In: *SoftwareX 24* (2023), p. 101551 [2].

  Resumen: Evolver es una herramienta basada en la formulación de la configuración automática y el diseño de metaheurísticas multiobjetivo como un problema de optimización multiobjetivo que puede resolverse utilizando el mismo tipo de algoritmos; es decir, estamos aplicando un enfoque de meta-optimización. Evolver proporciona implementaciones altamente configurables de algoritmos de optimización multiobjetivo representativos que pueden configurarse automáticamente a partir de una serie de problemas multiobjetivo utilizados como conjunto de entrenamiento y una lista de indicadores de calidad que son los objetivos a optimizar. Nuestra herramienta se basa en el framework jMetal, por lo que se pueden utilizar una gran cantidad de algoritmos existentes como meta-optimizadores.

Una interfaz gráfica permite a los científicos definir fácilmente escenarios de autoconfiguración, simplificando así el complejo proceso de encontrar configuraciones de algoritmos de alta calidad.

- José F. Aldana-Martín, María del Mar Roldán-García, Antonio J. Nebro, José F. Aldana-Montes. **"MOODY: an ontology-driven framework for standardizing multi-objective evolutionary algorithms"**. In: *Information Sciences 661* (2024), p. 120184 [3].

  Resumen: La aplicación de tecnologías semánticas, particularmente ontologías, en el ámbito de los algoritmos evolutivos multiobjetivo no es extendido a pesar de su eficacia en la representación del conocimiento. En este artículo presentamos MOODY, una ontología diseñada específicamente para formalizar este tipo de algoritmos, abarcando sus respectivos parámetros y problemas de optimización multiobjetivo basados en una caracterización de los *landscape*s de su espacio de búsqueda. MOODY está diseñado para ser particularmente aplicado en la configuración automática de algoritmos, que implica la búsqueda de los parámetros de un algoritmo para optimizar su rendimiento. En este contexto, observamos una notable ausencia de componentes y parámetros estandarizados, y consideraciones relacionadas como las caracteristicas de los problemas y las configuraciones de algoritmos. Esta falta de estandarización introduce dificultades en la selección de combinaciones de componentes válidas y en la reutilización de configuraciones algorítmicas entre diferentes implementaciones de algoritmos. MOODY ofrece un medio para infundir anotaciones semánticas en las configuraciones encontradas por herramientas automáticas, permitiendo la consulta eficiente de los resultados y la integración entre diversas fuentes mediante un grafo de conocimiento. Validamos nuestra propuesta presentando cuatro estudios de caso.

- José F. Aldana-Martín, Antonio J. Nebro, Juan J. Durillo, María del Mar Roldán García. **"A Study About Meta-Optimizing the NSGA-II Multi-Objective Evolutionary Algorithm"**. In: *9th International Conference on Metaheuristics and Nature Inspired Computing (In press)*. META 2023. Communications in Computer and Information Science. Springer, Cham [4].

  Resumen: El diseño automático de metaheurísticas multiobjetivo es una línea de investigación activa destinada a, dado un conjunto de problemas utilizados como conjunto de entrenamiento, encontrar la configuración de un optimizador multiobjetivo capaz de resolverlos eficientemente. El resultado esperado es que el algoritmo autoconfigurado pueda ser utilizado para encontrar aproximaciones precisas del frente de Pareto para otros proble-

mas. En este artículo, realizamos un estudio sobre la meta-optimización del conocido algoritmo NSGA-II, es decir, pretendemos usar NSGA-II como una herramienta de configuración automática para encontrar configuraciones de NSGA-II. Esta búsqueda se puede formular como un problema multiobjetivo donde las variables de decisión son los componentes y parámetros de NSGA-II y los objetivos son indicadores de calidad que deben minimizarse. Para desarrollar este estudio, confiamos en el framework jMetal. El análisis que proponemos tiene como objetivo responder a las siguientes preguntas de investigación: RQ1 - ¿como de complejo es construir el paquete de meta-optimización?, y RQ2 - ¿se pueden encontrar configuraciones precisas? Realizamos una experimentación para responder a estas preguntas.

Esta tesis doctoral no sólo contribuye a la literatura científica, sino que también enfatiza las aplicaciones prácticas, proporcionando una implementación de todos los resultados de esta investigación a través de múltiples repositorios de código abierto alojados bajo una organización dedicada a esta tesis en GitLab [1]. Los proyectos asociados estan disponibles para el publico bajo la licencia MIT (excepto los pesos de los modelos que se encuentran bajo Apache 2) e incluyen:

- *moody*: Este proyecto proporciona el modelo semántico desarrollado a través de esta tesis doctoral, implementado como una ontología OWL 2. Además, el repositorio de código incluye ejemplos en Python para anotar datos de acuerdo con el modelo, documentación HTML de la ontología y un pipeline de CI/CD que despliega la última versión como páginas de Gitlab. Tambien existe una URL permanente para la ontología, que respeta la negociación de contenido y devuelve tanto la documentación HTML como la ontología en formato RDF.
  - Repositorio Git: `https://gitlab.com/jfaldanam-phd/moody`
  - URL permanente: `https://w3id.org/moody`
- *Evolver*: Parte de la familia jMetal, *Evolver* es una herramienta para la configuración automática y el diseño de metaheurísticas multiobjetivo. Está desarrollado en Java 17 como un proyecto Maven y se basa en jMetal 6.1, pero también se proporciona como una imagen Docker desde el Registro de Contenedores de Github. Además, proporciona una interfaz gráfica de usuario opcional para facilitar su uso implementada en Python 3.9. El repositorio incluye: ambos proyectos, documentación sobre cómo usar la herramienta y un pipeline de CI/CD para construir automáticamente las nuevas versiones para ambos proyectos.

[1] `https://gitlab.com/jfaldanam-phd`

  - Repositorio Git: `https://github.com/jMetal/Evolver`

- *moorphology*: Biblioteca de software para caracterizar el *landscape* del espacio de variables y objetivos de un problema multiobjetivo continuo, basada en una implementación de jMetal. Implementado en Java 17 como un proyecto Maven, este proyecto proporciona un pipeline de CI/CD para construir automáticamente la última versión y desplegar la documentación javadocs de la biblioteca. Además, para evaluar esta biblioteca se proporciona una implementación de los problemas COCO [63, 20, 21] para jMetal.

  - Repositorio Git: `https://gitlab.com/jfaldanam-phd/moorphology`
  - Javadocs: `https://jfaldanam-phd.gitlab.io/moorphology/`
  - Repositorio de la suite *bbob-biobj* de COCO para jMetal: `https://gitlab.com/jfaldanam-phd/coco4jmetal`

- *SyntheticAI*: Generador sintético de problemas multiobjetivo, tanto en lenguaje natural como en el *framework* de jMetal. Este generador aprovecha LLMs y la tecnica de *few-shot learning* y está implementado en Python a través de la API de OpenAI.

  - Generador de problemas para generar el conjunto de datos de entrenamiento: `https://gitlab.com/jfaldanam-phd/syntheticai`

- *moostral*: Herramienta de software para la implementación automática de problemas de optimización multiobjetivo en el *framework* jMetal, basada por una versión adaptada de Mistral-7B. Esta herramienta está implementada en Python y utiliza las bibliotecas transformer y PEFT para la adaptación del modelo y streamlit para la interfaz gráfica de usuario.

  - Herramienta de implementación automática y código de adaptación: `https://gitlab.com/jfaldanam-phd/moostral`
  - Pesos para el LLM adaptado: `https://huggingface.co/jfaldanam/moostral-7B`

- *recommoonder*: Proporciona a los usuarios finales recomendaciones sobre metaheurísticas para resolver un problema específico, basado en conocimiento previo. Desarrollado en Python 3.10, incluye módulos para la ingestión, visualización, consulta y exportación de la información disponible en el grafo de conocimiento.

  - Repositorio Git: `https://gitlab.com/jfaldanam-phd/recommoonder`

- *meta-qdo*: Enfoque alternativo a *Evolver* mediante la aplicación de *Quality Diversity Optimization* [25] al proceso de autoconfiguración de un algoritmo multiobjetivo (ver Sección 5.9 para obtener más información). Es una implementación temprana para investigación, utilizando pyribs [150] y el algoritmo CMA-ES con margenes [60].
  - Repositorio Git: `https://gitlab.com/jfaldanam-phd/meta-qdo`

Conclusiones

A continuación, se proporcionan algunas conclusiones sobre cada una de las áreas que se han discutido.

Modelado semántico en el dominio de la optimización multiobjetivo

Esta tesis de doctoral propone un *framework* basado en ontologías para la estandarización en el campo de la optimización multiobjetivo, centrándose en algoritmos evolutivos. *moody*, la ontología principal que guía el *framework*, abarca aspectos desde la formalización de algoritmos evolutivos y sus parámetros hasta problemas multiobjetivo con sus características de *landscape*.

Eñ *moody* está desarrollado en OWL 2 y está enlazado a ontologías externas como BIGOWL, OPTION y DMOP. Se proporciona una implementación de referencia para añadir configuraciones de algoritmos evolutivos, convirtiendo la salida de la herramienta de autoconfiguración irace [103] al formato estándar RDF.

Se han proporcionado cuatro casos de uso para validar *moody*. Estos casos de uso son la mejora de herramientas de autoconfiguración a través del *framework moody*, la integración de datos de herramientas de autoconfiguración u otras fuentes, la consulta del grafo de conocimiento para obtener información útil y la exportación de este conocimiento a marcos de optimización utilizados en aplicaciones del mundo real, como pagmo [14] de la Agencia Espacial Europea.

Para proporcionar características del *landscape* de problemas multiobjetivo al grafo de conocimiento propuesto, se proporciona una implementación de estas métricas como el proyecto de software *moorphology*. Para validar el conjunto seleccionado de métricas, se evalua la estabilidad de las características sobre el conjunto biobjetivo COCO, acompañada de una implementación en jMetal.

Además, se estudia la calidad del conjunto de características para caracterizar problemas de optimización multiobjetivo.

Autoconfiguración de metaheurísticas

En esta tesis se presenta un estudio en el que el algoritmo NSGA-II se utiliza como meta-optimizador, es decir, como una herramienta que, dada un conjunto de problemas como conjunto de entrenamiento, tiene como objetivo encontrar las mejores configuraciones a partir de un conjunto de parámetros y componentes de NSGA-II. Utilizando un esquema de codificación simple y aprovechando características existentes en jMetal, la propuesta se desarrolla completamente dentro de jMetal, eliminando así la necesidad de cualquier herramienta externa.

Se han definido varios experimentos para validar esta propuesta considerando dos escenarios y tres experimentos para cubrir tanto la búsqueda automática de configuraciones de NSGA-II para conjuntos de entrenamiento de uno o varios problemas. Los resultados de estos experimentos revelan que el meta-optimizador es capaz de encontrar configuraciones de NSGA-II que logran con éxito los objetivos definidos.

Después de validar el enfoque, se presenta *Evolver* como un paquete de software destinado a la meta-optimización de metaheurísticas multiobjetivo. Al definir la búsqueda automática de configuraciones para optimizadores multiobjetivo como un problema multiobjetivo, *Evolver* facilita al usuario encontrar variantes de algoritmos que se adaptan a varios problemas de optimización utilizados como conjunto de entrenamiento. Esta herramienta ofrece una serie de metaheurísticas multiobjetivo representativas que son altamente configurables (NSGA-II, MOEA/D, SMS-EMOA, MOPSO).

Evolver está implementado en Java y se basa en el *framework* jMetal, por lo que se pueden utilizar una gran cantidad de metaheurísticas ya existentes como meta-optimizadores. Los usuarios familiarizados con jMetal se sentirán cómodos con *Evolver* y tendrán la oportunidad de utilizarlo como herramienta de investigación en la línea de configuración automática de metaheurísticas. La interfaz grafica proporcionada permite a los usuarios no expertos configurar y ejecutar fácilmente una ejecución de meta-optimización.

El funcionamiento de *Evolver* se ilustra considerando un ejemplo que representa un escenario típico en el que un ingeniero pretende encontrar una variante de NSGA-II para resolver un tipo determinado de problemas.

Además, se propone un novedoso enfoque alternativo mediante la aplicación de *Quality-Diversity* para la configuración automática de metaheurísticas.

La aplicación de estas técnicas de optimización proporciona un conjunto de configuraciones algorítmicas diversas que se combinan para mejorar la robustez y la generalización de las configuraciones individuales, y se evalúa sobre un gran conjunto de problemas de referencia.

Implementación automática de problemas de optimización multiobjetivo

En este trabajo se aborda el desafío de convertir la representación en lenguaje natural o textual de la formulación de problemas de optimización multiobjetivo en código ejecutable, diseñando una herramienta que automatiza el proceso de implementación.

Las principales contribuciones radican en la adaptación de un LLM sobre un conjunto de datos sintéticos de problemas multiobjetivo generados por un generador de problemas personalizado proporcionado junto con el modelo. El modelo propuesto traduce de manera efectiva la representación textual de la formulación de un problema de optimización en su implementación equivalente en el *framework* jMetal, y el modelo se valida empíricamente utilizando una serie de diez problemas multiobjetivo del mundo real.

Además, el modelo entrenado se incrusta dentro de una herramienta con una interfaz gráfica de usuario y un conjunto de pasos de validación para garantizar la corrección de la implementación proporcionada sin requerir la alta potencia computacional generalmente asociada con los LLM. Tanto los pesos del modelo como la herramienta están disponibles bajo una licencia abierta.

Recomendación automática de algoritmos de optimización multiobjetivo

La contribución final es el desarrollo de una herramienta llamada *recommoonder*. Esta herramienta integra las capacidades de las herramientas anteriores en un motor de recomendación en un paquete fácil de usar para el usuario. *recommoonder* esta diseñado para ayudar a los usuarios no expertos a seleccionar configuraciones de algoritmos que superen las configuraciones predeterminadas estándar.

Posteriormente, la herramienta se evalúa en problemas conocidos y desconocidos, y se confirma empíricamente cómo las medidas de similitud que se describen en el Capítulo 4 son relevantes para la recomendación de configuraciones algorítmicas específicas.

Trabajos futuros

Esta sección enumera líneas potenciales de trabajo para continuar la investigación presentada en esta tesis doctoral. El código fuente de todas las herramientas presentadas en esta tesis es de código abierto bajo una licencia permisiva para cualquier persona interesada en continuar cualquiera de las líneas de investigación presentadas.

- La ontología *moody* puede extenderse incluyendo otras metaheurísticas, como la optimización por enjambre de partículas, y nuevos problemas, incluyendo problemas de optimización discreta. En este sentido, trabajar con problemas del mundo real es particularmente interesante, pero plantea un desafío especial, ya que usualmente el frente de Pareto es desconocido.

- Un análisis en profundidad sobre si el conjunto de características del *landscape* presentado en esta tesis es el conjunto óptimo de características, cómo afectan al rendimiento algorítmico y un análisis cuantitativo sobre cómo se relacionan con la similitud de problemas son líneas de investigación abiertas.

- La métrica propuesta para calcular la similitud entre problemas de optimización multiobjetivo, que actualmente se basa en medir la distancia entre sus características del *landscape*, puede mejorarse mediante la integración de técnicas de aprendizaje automático. Estas técnicas podrían emplearse para asignar dinámicamente pesos variables a cada característica, dependiendo, por ejemplo, del algoritmo utilizado.

- Otra línea interesante implica un estudio más profundo sobre la similitud entre los problemas de una misma familia, qué características afectan esa similitud y los sesgos introducidos por los autores.

- En el tema de la autoconfiguración, un estudio que examine hasta qué punto se puede reducir el número de evaluaciones del metaoptimizador en la búsqueda mientras que las configuraciones resultantes de NSGA-II aún pueden resolver los problemas de manera eficiente. Un estudio relacionado es encontrar hasta qué punto se puede reducir el límite computacional para las configuraciones que se están evaluando mientras aún se obtienen buenas configuraciones. Esta última opción es especialmente interesante en problemas computacionalmente costosos.

- Adaptar *Evolver* para soportar la optimización en problemas con un frente de Pareto desconocido. Esto implica el uso de métricas de calidad para el enfoque de metaoptimización que no requieran un frente de referen-

cia, como el hipervolumen, y ajustar dinámicamente el frente de referencia durante el proceso de búsqueda. Esta segunda opción implica usar un archivo externo de soluciones que serán reevaluadas al actualizar el frente de referencia.

- En la implementación automática de problemas de optimización multiobjetivo, el entrenamiento de LLMs para generar salida para otros *framework* de optimización como PlatEMO o Pymoo es una línea de investigación abierta. Esta extensión en PlatEMO abriría la herramienta a un mayor número de usuarios. Otras líneas de investigación para mejorar la eficiencia del modelo son la cuantización del modelo o el preentrenamiento con tokenizadores personalizados diseñados para reconocer símbolos específicos del *framework*. Esta última permitiría reducir el número de tokens necesarios para generar código válido, mejorando tanto la latencia como el coste de inferencia.
- Un enfoque diferente para mejorar el LLM propuesto es el estudio sobre el uso de tecnologías semánticas para inyectar conocimiento de dominio en el modelo, con la intención de generar implementaciones de problemas a partir de descripciones informales en lenguaje natural o más cercanas al lenguaje utilizado por los humanos.
- Continuar la investigación sobre el uso de *Quality Diversity Optimization* con un estudio sobre la aplicación de conjuntos diversos en problemas del mundo real también es una línea de investigación abierta.

Chapter 1

Introduction

This chapter provides an overview of the primary fields explored in this PhD thesis, offering insights into the motivation behind this research, along with its principal goals and objectives. This chapter concludes by highlighting the primary contributions of this PhD to the related fields, along with outlining the organization of the remaining chapters of the thesis.

1.1 Background

Optimization has been an integral part of human problem-solving for centuries, predating even the oldest of known historical figures. These kind of problems, which involve finding the best solution from all feasible solutions, have been faced in various forms since ancient times. For instance, in ancient Greece, Euclid laid the foundation for geometric optimization by defining the Euclidean distance, which represents the minimum distance between two points as a straight line. This insight from Euclid can be framed as an optimization problem where the objective is to minimize the distance between two points. By formulating it in this way, mathematical techniques can be applied to find the optimal solution, which turns out to be the straight line connecting the two points. This concept of optimization, rooted in the fundamental principles of geometry, has transcended time and disciplines, becoming a cornerstone in fields ranging from mathematics and physics to engineering and computer science.

Until the 1980s, optimization techniques were mostly classical mathematical methods. The 1980s marked a significant turn in the field with the emergence of evolutionary algorithms [28, 33]. These algorithms, inspired by biological evolution, were particularly adept at solving complex optimization problems that were previously intractable. They represent stochastic techniques, which, while not guaranteeing the identification of the global optimum for problems, frequently yield quasi-optimal solutions within a reasonable time frame. Remarkably, these algorithms do not necessarily demand specific domain information about the problem being optimized to generate robust approximations to the optimal solution. These characteristics contribute to their widespread adoption, making them highly utilized techniques in various domains.

During the 1990s, evolutionary algorithms, became part of a broader category known as metaheuristics. These high-level techniques operate by manipulating lower-level methods, often heuristics, to deliver efficient optimization algorithms. This integration expanded the scope of optimization approaches, encompassing various algorithms, such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Tabu Search (TS), and more [17]. Over the last two decades, a focus of research has intensified on the study of multi-objective optimization problems and their resolution using metaheuristics. These problems, frequently encountered in real-world applications across various domains, are characterized by being formulated based on two or more conflicting functions that must be optimized simultaneously. This implies that improving one of them leads to a worsening in the rest [45].

The challenge of multi-objective optimization lies in navigating the contra-

dictory nature of these objectives, where the optimal solution isn't a singular outcome but rather a set of compromise solutions, known as the Pareto set. In this set, no solution surpasses another across all objectives, and its correspondence in the objective space is named as Pareto front. When applied to solve multi-objective problems, metaheuristics are aimed then at finding an accurate approximation to the Pareto front.

The challenging landscape of optimization is further defined by the No Free Lunch theorem [160], which posits that there is no algorithm universally optimal for solving all optimization problems within a class. This implies that, when faced with a specific optimization problem, the optimal algorithm to address it cannot be predetermined. The intricate nature of problem-specific nuances demands a dynamic and adaptable approach to algorithm selection. At the same time, metaheuristics are extremely sensitive to their configuration parameters, relying heavily on parameter tuning to boost the effectiveness of these techniques. This requires a deep understanding of the field and, more importantly, the specific algorithm used.

End-users (biologists, engineers, economists, etc.) who need to optimize multi-objective problems are often not experts in metaheuristics, and often utilize the default configuration of a popular algorithm without tuning its parameters to their real-world problem. Therefore, a current and active research area involves developing tools that enable these users to easily find an algorithm (and configuration) capable of effectively solving their problems. In this regard, techniques are being investigated for the automatic tuning of metaheuristics parameters [73, 122] and even for the automatic design of algorithms [13]. These techniques aim to adapt the parameters of metaheuristics automatically, optimizing their performance for specific problems. However, they come with the downside of requiring substantial computational resources. This area of research is continuously evolving, seeking to bridge the gap between the complexity of real-world problems and the effectiveness of optimization techniques.

In this context, this PhD thesis proposes an alternative approach based on using existing knowledge to create a tool to provide end-users recommendations [161] of superior configurations of algorithms beyond their default settings. Building on this idea, this work leverages the established effectiveness of semantic technologies, which have been proven in many other fields. These technologies are pivotal in integrating and representing domain knowledge, supporting data standardization, and facilitating semantic integration from multiple sources [134]. Ontologies are the most widely spread method to describe the knowledge, especially allowing a formal and logic-based definition of concepts as a common vocabulary to share information in a specific domain [145]. To power

the recommendations, ontologies are used to define the structure of a knowledge graph where the results of configuration generation tools can be stored. To further facilitate end-users the use of metaheuristics, Large Language Models (LLMs) [171] are studied as a technology to help with the implementation of multi-objective optimization problems within optimization frameworks.

1.2 Motivation

The main hypothesis underpinning this PhD thesis proposal is that given previous knowledge on the relationship between a specific algorithmic configuration and the quality of the result of said algorithm solving a problem and given a similitude metric between two problems, it is possible to provide recommendations to non-expert users to choose an algorithmic configuration to efficiently solve a specific problem.

The use of a knowledge-based approach is key in this work. Ontologies allow for the unambiguous description of entities and the relationships among these entities in an application domain. Defining a semantic model based on an OWL 2 (Web Ontology Language) ontology [58] for the automatic configuration of optimization algorithm parameters enables us to exploit Semantic Web technologies.

The use of Semantic Web technologies allows not only to formally describe problems, algorithms, parameters, configurations, and quality metrics but also to annotate the results of configuration generation tools and integrate all this information easily in a knowledge graph. Furthermore, ontologies allow us to define logical constraints in the domain, enabling us to verify the validity of configurations, facilitates querying the knowledge graph and offers the possibility to apply semantic reasoning over it [71].

This research line is framed within the context of the “Aether-UMA: Una aproximación holística de smart data para el análisis de datos guiado por el contexto: explotación de semántica y contexto” (PID2020-112540RB-C41) project, funded by the Spanish Ministry of Science and Innovation, as its objectives are strongly aligned with it.

1.3 Objectives

The main objective of this thesis is the design and development of a recommendation tool for metaheuristics for solving multi-objective continuous optimization problems by applying a knowledge-based approach based on Semantic Web

technologies. This objective is split in several smaller ones as follows:

Objective 1: Design and implement an OWL 2 ontology to represent multi-objective optimization knowledge, for both the algorithms and problems, to serve as the semantic model to generate a Knowledge graph.

- 1.1: Select a representative set of multi-objective optimization algorithms to model their parameters.
- 1.2: Selection of a set of benchmark multi-objective problems and the characteristics to study over them.
- 1.3: Design and implement the ontology.

Objective 2: Automatic generation of configurations of the previously chosen algorithms and problems.

- 2.1: Evaluate a meta-optimization approach for the generation and evaluation of algorithmic configurations.
- 2.2: Design and implement the annotation methodology to add configurations and problem characteristics to the knowledge graph.

Objective 3: Design a set of SPARQL queries to search, for a given problem, the algorithms and configurations more promising to solve it.

- 3.1: Design the SPARQL queries to provide users recommendations.
- 3.2: Experimentation and validation.

Objective 4: Define or select a metric to evaluate the similitude between several multi-objective problems, according to their landscape characteristics.

- 4.1: Extraction of the landscape characteristics of multi-objective problems.
- 4.2: Design of a metric from the available characteristics to define the similitude between several problems according to their landscape characteristics.
- 4.3: Experimentation and validation of the metrics chosen.

Objective 5: Implement auxiliary tools to facilitate multi-objective optimization problem implementation within a supported framework for the recommendation system.

- 5.1: Evaluate the use of existing LLMs for the automatic implementation of multi-objective optimization problems from a textual representation.

5.2: Generate a high-quality dataset of synthetic multi-objective problems to enable the training of a model.

5.3: Fine-tune a LLM to solve this task within the computing capacities of a single user.

5.4: Evaluate the fine-tuned model over a set of real-world problems.

Objective 6: Design and implementation of a recommendation tool

6.1: Implement the recommendation tool.

6.2: Testing and validation of the recommendations provided.

Additionally, as a non-functional objective, the goal is to provide open source implementations of all contributions related to this PhD thesis, following the best practices in software development and providing high-quality documentation for each project. This approach facilitates the utilization of these implementations, both within and beyond the research community.

1.4 Thesis contributions

The main contributions of this PhD thesis are related with the objectives described in the previous section as follows:

- An ontology driven semantic framework to consolidate multi-objective optimization knowledge in the form of an OWL 2 ontology following the FAIR principles (Findable, Accessible, Interoperable, and Reusable) [157]. Chapter 3 presents the *moody* (Multi-Objective Optimization ontologY) ontology, the semantic backbone of the semantic framework addressing **Objectives 1** and **3**.
- In Chapter 4, a study is conducted on the landscape of the search space of the selected multi-objective problems (**Objective 4**). A set of metrics has been selected that will be used to characterize the landscape of the variable and objective space, and implement *moorphology*, a software library to calculate these metrics on problems implemented on the jMetal framework.
- Chapter 5 covers **Objective 2** and reviews the state of the art in auto-configuration for multi-objective optimization and proposes a tool aimed at automatically configuring and designing metaheuristics for multi-objective optimization problems, named *Evolver*. The key of *Evolver* is to formulate this process as a multi-objective optimization problem, which allows the

application of well-established multi-objective optimization algorithms to find targeted configurations for metaheuristics.

- An evaluation of state-of-the-art LLMs for solving the task described in **Objective 5** is presented in Chapter 6. Following the evaluation, the chapter focuses on the rest of the objectives: developing a tool powered by an LLM for the automatic implementation of multi-objective optimization problems is described, including the generation of a training dataset, the fine-tuning process and subsequent evaluation of the model.

- **Objective 6** is addressed in Chapter 7, developing *recommoonder*, a recommendation tool that integrates the other contributions of this PhD into a cohesive recommendation engine to provide users recommendations of superior configurations of algorithms beyond their default settings.

A visual overview of this contributions can be seen in Figure 1.1.

The development of this PhD has resulted in the publication of two manuscripts in high-impact journals, indexed in the Journal Citation Reports (JCR), and one other article presented on an international conference:

- José F. Aldana-Martín, Juan J. Durillo, Antonio J. Nebro. **"Evolver: Meta-optimizing multi-objective metaheuristics"**. In: *SoftwareX 24* (2023), p. 101551 [2].

  Abstract: Evolver is a tool based on the formulation of the automatic configuration and design of multi-objective metaheuristics as a multi-objective optimization problem that can be solved by using the same kind of algorithms; i.e., we are applying a meta-optimization approach. Evolver provides highly configurable implementations of representative multi-objective solvers which can be automatically configured from a number of multi-objective problems used as the training set and a list of quality indicators which are the objectives to be optimized. Our tool is based on the jMetal framework, so a large number of existing algorithms can be used as meta-optimizers. A graphical user interface allows scientists to easily define auto-configuration scenarios, thus simplifying the complex process of finding high-quality algorithm settings.

- José F. Aldana-Martín, María del Mar Roldán-García, Antonio J. Nebro, José F. Aldana-Montes. **"MOODY: an ontology-driven framework for standardizing multi-objective evolutionary algorithms"**. In: *Information Sciences 661* (2024), p. 120184 [3].

  Abstract: The application of semantic technologies, particularly ontologies,

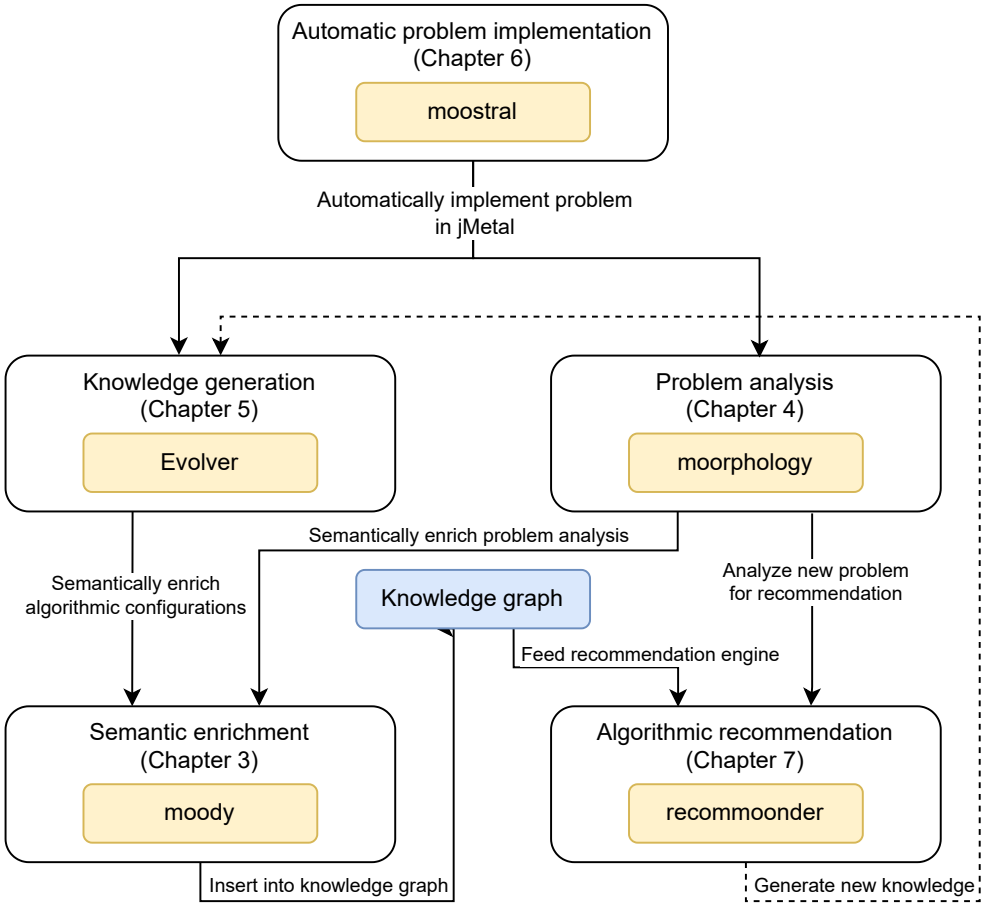


Figure 1.1: Global overview of the contributions of this PhD thesis.

in the realm of multi-objective evolutionary algorithms is overlook despite their effectiveness in knowledge representation. In this paper, we introduce MOODY, an ontology specifically tailored to formalize these kinds of algorithms, encompassing their respective parameters, and multi-objective optimization problems based on a characterization of their search space landscapes. MOODY is designed to be particularly applicable in automatic algorithm configuration, which involves the search of the parameters of an optimization algorithm to optimize its performance. In this context, we observe a notable absence of standardized components, parameters, and related considerations, such as problem characteristics and algorithm configurations. This lack of standardization introduces difficulties in the

selection of valid component combinations and in the re-use of algorithmic configurations between different algorithm implementations. MOODY offers a means to infuse semantic annotations into the configurations found by automatic tools, enabling efficient querying of the results and seamless integration across diverse sources through their incorporation into a knowledge graph. We validate our proposal by presenting four case studies.

- José F. Aldana-Martín, Antonio J. Nebro, Juan J. Durillo, María del Mar Roldán García. **"A Study About Meta-Optimizing the NSGA-II Multi-Objective Evolutionary Algorithm"**. In: *9th International Conference on Metaheuristics and Nature Inspired Computing (In press)*. META 2023. Communications in Computer and Information Science. Springer, Cham [4].

  Abstract: The automatic design of multi-objective metaheuristics is an active research line aimed at, given a set of problems used as training set, to find the configuration of a multi-objective optimizer able of solving them efficiently. The expected outcome is that the auto-configured algorithm can be used of find accurate Pareto front approximations for other problems. In this paper, we conduct a study on the meta-optimization of the well-known NSGA-II algorithm, i.e., we intend to use NSGA-II as an automatic configuration tool to find configurations of NSGA-II. This search can be formulated as a multi-objective problem where the decision variables are the NSGA-II components and parameters and the the objectives are quality indicators that have to be minimized. To develop this study, we rely on the jMetal framework. The analysis we propose is aimed at answering the following research questions: RQ1 - how complex is to build the meta-optimization package?, and RQ2 - can accurate configurations be found? We conduct an experimentation to give an answer to these questions.

This PhD thesis not only contributes to scientific literature but also emphasizes practical applications, providing an implementation of all research findings through multiple open source repositories hosted under a dedicated GitLab organization for this thesis [1]. The associated projects are licensed to the public under the MIT license (model weights are release under Apache 2) and include:

- *moody*: This project provides the semantic model develop through this PhD thesis, implemented as a OWL 2 ontology. Additionally, the code repository includes examples in Python to annotate data according to the model, html documentation of the ontology and a CI/CD pipeline that deploys the latest version as Gitlab pages. It provides a permanent URL for the ontology, that

[1] `https://gitlab.com/jfaldanam-phd`

respects content negotiation and returns either the HTML documentation or the ontology in RDF format.

  - Git repository: `https://gitlab.com/jfaldanam-phd/moody`
  - Permanent URL: `https://w3id.org/moody`

- *Evolver*: Part of the jMetal family, *Evolver* is a tool for the automatic configuration and design of multi-objective metaheuristics. It is developed in Java 17 as a Maven project and is based in jMetal 6.1, but is also provided as a Docker image from the Github Container Registry. Additionally, it provides an optional Graphical User Interface for ease of use implemented in Python 3.9. The repository includes: both projects, documentation on how to use the tool and a CI/CD pipeline to automatically build the new releases for both projects.
  - Git repository: `https://github.com/jMetal/Evolver`
- *moorphology*: Software library to characterize the landscape of the variable and objective space of a continuous multi-objective problem, based on a jMetal implementation. Implemented in Java 17 as a Maven project, this project provides CI/CD pipeline to automatically build the latest release, as well as deploy the javadocs documentation of the library. Additionally, to evaluate this library an implementation of bindings of the COCO problems [63, 20, 21] for jMetal is provided.
  - Git repository: `https://gitlab.com/jfaldanam-phd/moorphology`
  - Java docs: `https://jfaldanam-phd.gitlab.io/moorphology/`
  - Git repository for COCO's *bbob-biobj* suite for jMetal: `https://gitlab.com/jfaldanam-phd/coco4jmetal`
- *SyntheticAI*: Synthetic generator for multi-objective problems, both in natural language and in the jMetal framework. This generator leverages LLMs and few-shot learning and is implemented in Python via OpenAI's API.
  - Problem generator to generate the training dataset: `https://gitlab.com/jfaldanam-phd/syntheticai`
- *moostral*: Software tool for the automatic implementation of multi-objective optimization problems into the jMetal framework, powered by a fine-tuned version of Mistral-7B. This tool is implemented in Python and uses the transformer and PEFT libraries for the fine-tuning of the model and streamlit for the graphical user interface.

  - Automatic implementation tool and fine-tuning code: `https://gitlab.com/jfaldanam-phd/moostral`
  - Weights for the fine-tuned LLM: `https://huggingface.co/jfaldanam/moostral-7B`

- *recommoonder*: Wrapper project to provide end users with recommendations on metaheuristics to solve a specific problem, based on previous knowledge. Developed in Python 3.10, it includes modules for the ingestion, visualization, query and export of the information available in the knowledge graph.
  - Git repository: `https://gitlab.com/jfaldanam-phd/recommoonder`
- *meta-qdo*: Quality-Diversity optimization for metaheuristics is an alternative approach to *Evolver* by applying Quality-Diversity Optimization [25] to the process of auto-configuration of a multi-objective metaheuristic (see Section 5.9 for further information). It is an early research implementation using pyribs [150] and the CMA-ES with margin algorithm [60].
  - Git repository: `https://gitlab.com/jfaldanam-phd/meta-qdo`

Chapter 2

Theoretical Foundation

The following chapter offers an overview of the main concepts and theoretical foundations in the fields of multi-objective optimization, auto-configuration of algorithms, large language models and Semantic Web technologies. These areas are closely linked to the research conducted in this PhD thesis, laying the groundwork for understanding its contributions.

2.1 Multi-objective optimization

As multi-objective optimization is the main field of study throughout this PhD thesis, this section starts by defining the concept of optimization.

Definition 1: Optimization problem

In mathematics, optimization is the process of finding the best solution or outcome from a set of possible alternatives. The quality of a solution is measured by an objective or fitness function defined as $f : S \to \mathbb{R}$, where $S \neq \emptyset$ represents the search space. To solve an optimization problem, the goal is to search for a solution i^* that satisfies Equation 2.1. The search space S in a optimization problem can be reduced by additional constraints, which may come from the problem's domain.

$$f(i^*) \leq f(i), \quad \forall i \in S, \quad (i^*) \in S \tag{2.1}$$

This process can be of maximization or minimization, depending on the nature of the optimization problem, but this does not change the process as a equality can be established between both types as shown in Equation 2.2 [57].

$$\max\{f(i) \mid i \in S\} \equiv \min\{-f(i) \mid i \in S\} \tag{2.2}$$

Equation 2.3 illustrate a simple optimization problem and its optimal solution is visually depicted in Figure 2.1.

$$\max_{x,y \in \mathbb{R}} \; -(x^2 + y^2) + 4 \tag{2.3}$$

Optimization problems are typically categorized based on the number of objectives being optimized. These classifications include mono-objective optimization, which focuses on a single objective; multi-objective optimization, which involves optimizing multiple conflicting objectives; and many-objective optimization, specifically addressing scenarios with four or more objectives. Multi-objective optimization is an area of research concerned with finding an optimal solution to a multi-objective optimization problem.

Definition 2: Multi-objective optimization problem

Formally, a multi-objective optimization problem can be defined as a problem of finding a vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ which minimizes the vector function $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T$ while it satisfies m inequality constraints

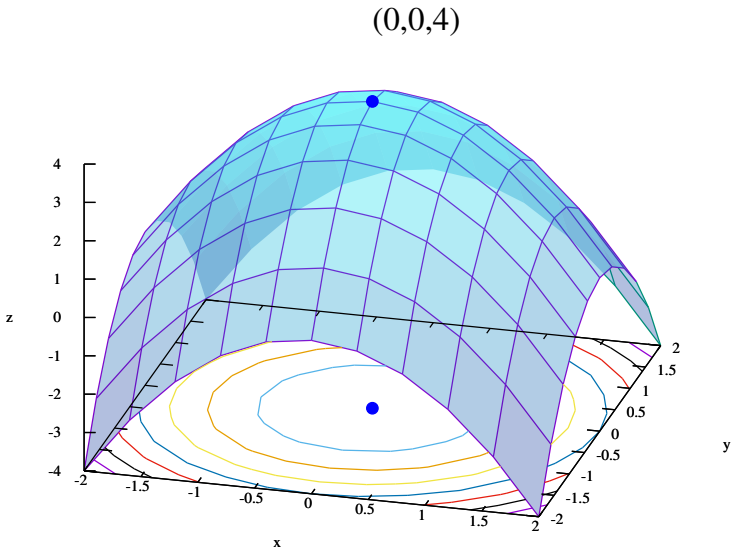


Figure 2.1: Graph of a surface defined by $f(x, y) = -(x^2 + y^2) + 4$. The optimum value is shown as a blue dot.

$g_i(\vec{x}) \geq 0$, $i = 1, 2, \ldots, m$ and p equality constraints $h_i(\vec{x}) = 0$, $i = 1, 2, \ldots, p$, where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables. According to the domain of the solutions, there are binary ($x_{1..n} \in \mathbb{B}$), integer ($x_{1..n} \in \mathbb{N}$) and continuous ($x_{1..n} \in \mathbb{R}$) optimization problems. More generally, optimization problems are often categorized in two big groups: continuous problems and discrete problems, which include integer and combinatorial (permutation, tree, graph...) optimization problems.

The solution to a multi-objective optimization problem is not a single solution, but a set of them, known as the Pareto optimal set. To better understand this, let's define the concepts of Pareto optimality, Pareto dominance and, finally, the Pareto optimal set or simply Pareto set.

Definition 3: Pareto optimality

The feasible region $\mathbf{\Omega}$ is defined by the set of all values satisfying the constraints, and any point $\vec{x} \in \mathbf{\Omega}$ is a feasible solution.

A point $\vec{x}^* \in \mathbf{\Omega}$ is Pareto Optimal if for every $\vec{x} \in \mathbf{\Omega}$ and $I = \{1, 2, \ldots, k\}$ either $\forall i \in I$, $f_i(\vec{x}) = f_i(\vec{x}^*)$ or there is at least one $i \in I$ such that $f_i(\vec{x}) > f_i(\vec{x}^*)$.

Plainly, this means that a vector $\mathbf{x}^*$ is Pareto optimal if there does not exist another feasible vector $\mathbf{x}$ which would improve some of the objectives without causing a worsening of at least one other criterion at the same time.

Definition 4: Pareto dominance

A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, as shown in Equation 2.4.

$$\vec{u} \preceq \vec{v} \iff \forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i \tag{2.4}$$

Definition 5: Weak Pareto dominance

A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to weakly dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq_w \vec{v}$) if and only if $\vec{u}$ is partially less than or equal to $\vec{v}$, as illustrated in Equation 2.5.

$$\vec{u} \preceq_w \vec{v} \iff \forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i \leq v_i \tag{2.5}$$

In weak Pareto dominance, the condition for dominance allows for equality in the comparison of corresponding components of the vectors.

Definition 6: Pareto optimal set

For a given multi-objective optimization problem $\vec{f}(\vec{x})$, the Pareto optimal set is the set of feasible solutions such as no other feasible solution dominates it, as shown in Equation 2.6.

$$P^* = \{\vec{x} \in \mathbf{\Omega} \mid \neg\exists \vec{x'} \in \mathbf{\Omega},\ \vec{f}(\vec{x'}) \preceq \vec{f}(\vec{x})\} \tag{2.6}$$

Definition 7: Pareto front

For a given multi-objective problem $\vec{f}(\vec{x})$ and its Pareto optimal set P^*, the Pareto front is formed by the same set of solutions, but defined in the objective space instead of the variable space. This is defined in Equation 2.7.

$$PF^* = \{\vec{f}(\vec{x}), \vec{x} \in P^*\}. \tag{2.7}$$

The objective of multi-objective optimization is to find the Pareto front of the multi-objective problem in question. Theoretically, a Pareto front could contain

an infinite amount of points, such as when solving continuous problems. In practice, the goal is to find an approximation to the Pareto front that only contains a finite number of points. For this practical limitation, it is important that the approximation is as close as possible to the true Pareto Front (convergence) and that the points in it are uniformly spread (diversity). Good convergence ensures that near optimal solutions have found, while good diversity means that the solutions have good coverage of the search space, leaving no regions unexplored. To measure convergence and diversity, quality indicators are used.

Definition 8: Quality Indicator

A quality indicator is a metric used to measure the quality of a solution to a optimization problem. Usually, they are used to measure the convergence and diversity of a solution against an approximation of the Pareto front of a given problem.

One key property of quality indicators is whether they are Pareto compliant. Pareto compliance is an order property that allows a quality indicator to not contradict the order imposed by the Pareto dominance relation. Weak (or partial) Pareto compliance is a variant that acknowledges situations where there may exist ties or ambiguities in dominance relationships between solutions. The main quality indicators used in this thesis are as follow:

- Inverted Generational Distance (IGD): The Generational Distance (GD) indicator [154] is used to measure how far the non-dominated solutions are from the Pareto optimal set, measuring both convergence and diversity. The GD is shown in Equation 2.8, where n is the number of non-dominated solutions and d_i is the Euclidean distance in the objective space between each solution and the nearest member of the Pareto optimal set. The Inverted Generational Distance [29] uses the same defintion, but using the Pareto Front as the reference, and comparing each of its elements to the solutions. As such, for a set A and a reference set R the IGD is calculated as $IGD(A, R) = GD(R, A)$.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2.8}$$

- Inverted Generational Distance Plus (IGD+): The Inverted Generational Distance Plus is a variation of the Inverted Generational Distance to adapt the metric to being partial Pareto compliant [78]. This is fixed by using a limited distance to avoid negative values, as shown in Equation 2.9 where s_i is an element of the solution front and z_i is the nearest member of the

Pareto optimal set.

$$d_i^+ = max\{s_i - z_i, 0\}, i = 1, 2, .., m \tag{2.9}$$

- Spread (Δ): The spread metric can be used to measure the diversity of the solution front [34]. Defined by Equation 2.10, where n is the size of the solution front, d_i the Euclidean distance between two consecutive solutions and $\bar{d}$ is the average of this distances. This metric has a value of 0 when there is a perfect distribution.

$$\Delta = \sum_{i=1}^{n} \frac{|d_i - \bar{d}|}{n} \tag{2.10}$$

- Epsilon (I_ϵ): The convergence metric epsilon [174] is based on the concept of ϵ-dominance, shown in Equation 2.11. Loosely speaking, a vector $\vec{v}$ ϵ-dominates ($\preceq_\epsilon$) a vector $\vec{u}$, if by multiplying each objective in $\vec{u}$ by ϵ, the resulting vector is still weakly dominated by $\vec{v}$.

  The binary epsilon indicator ($I_\epsilon(A, B)$) compares two solution fronts (usually the solution obtained against the Pareto front) by the minimum factor ϵ, such that any objective vector in B is ϵ-dominated in at least one objective vector in A. The binary epsilon indicator is formally describe in Equation 2.12.

$$\vec{u} \preceq_\epsilon \vec{v} \iff \forall 1 \leq i \leq n : v_i \leq \epsilon \cdot u_i \wedge \epsilon > 0 \tag{2.11}$$

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall \vec{u} \in B \, \exists \vec{v} \in A : \vec{v} \preceq_\epsilon \vec{u}\} \tag{2.12}$$

  Note, that in this PhD thesis the additive epsilon indicator variation will be used, defined by adding instead of multiplying ϵ to the dominated vector, as shown in Equations 2.13 and 2.14.

$$\vec{u} \preceq_{\epsilon+} \vec{v} \iff \forall 1 \leq i \leq n : v_i \leq \epsilon + u_i \wedge \epsilon > 0 \tag{2.13}$$

$$I_{\epsilon+}(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall \vec{u} \in B \, \exists \vec{v} \in A : \vec{v} \preceq_{\epsilon+} \vec{u}\} \tag{2.14}$$

- Hypervolume (HV): The hypervolume is a joint metric for convergence and diversity [173] that is defined by calculating the n-dimensional space covered by a solution front $\mathbf{X}$ with respect to reference point (usually formed by the worst known value for each objective). Equation 2.15 formally defines this where $\vec{x}^{ref}$ refers to a chosen reference point and λ refers to the Lebesgue measure [88] in a problem with M objectives.

$$HV(\vec{x}^{ref}, \mathbf{X}) = \lambda \left(\bigcup_{\mathbf{X}_n \in \mathbf{X}, 1 \leq k \leq M} \left[f_k(\mathbf{X}_n), x_k^{ref} \right] \right) \tag{2.15}$$

- Normalized hypervolume (NHV): The normalize hypervolume [174] is a variant of HV, defined as 1.0 minus the hypervolume of the front divided by the hypervolume of a reference front, both with respect the same reference point, as shown in Equation 2.16. Note that the NHV is not defined if the hypervolume of the reference front is 0.

$$NHV(\vec{x}^{ref}, \mathbf{X^{ref}}, \mathbf{X}) = 1 - \frac{HV(\vec{x}^{ref}, \mathbf{X})}{HV(\vec{x}^{ref}, \mathbf{X^{ref}})} \tag{2.16}$$

From the mentioned quality indicators, the hypervolume, and its normalized variant, is the only one that does not require previous knowledge of the Pareto front as a reference to be measure convergence, only requiring a reference point, and is the only indicator that is fully Pareto compliant. It is important to note that normalizing the reference front and the obtained solution front is required to avoid misleading results with all the quality indicators [65].

2.2 Metaheuristics

Now let's move the focus about how to obtain the Pareto front for a multi-objective problem. Due to impracticability of obtaining the Pareto optimal set, non-exact techniques like metaheuristics [46] are often used in many situations, such as when solving real world problems. These constitute a broad family of optimization algorithms that can get quasi-optimal solutions by searching the solution space of the problem in a reasonable time.

Definition 9: Metaheuristic

A metaheuristic can be defined as a high level strategy to guide a set of underlying heuristics by combining different concepts for the exploration and exploitation of the search space in order to find a balance between diversification

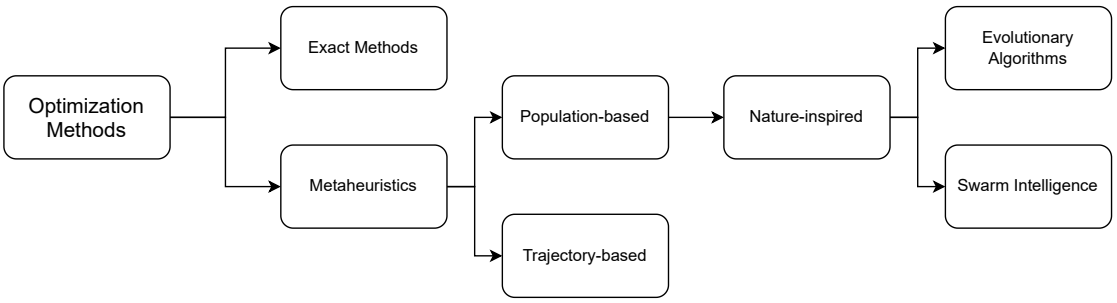


Figure 2.2: Classification of the different kind of optimization methods, focusing on the relevant branches for this thesis.

```
P(0) ← GenerateInitialPopulation()
t ← 0
Evaluation(P(0))
while not TerminationCriterionIsMet() do
  P'(t) ← Selection(P(t))
  Q(t) ← Variation(P'(t))
  Evaluation(Q(t))
  P(t+1) ← Replacement(P(t), Q(t))
  t ← t+1
end while
```

Listing 2.1: Pseudo-code of an evolutionary algorithm.

and intensification [17, 55]. This intelligent process allows to find near-optimal solutions efficiently. A classification of the different metaheuristics can be found in Figure 2.2.

Definition 10: Evolutionary algorithm

An evolutionary algorithm is inspired by biological evolution, using mechanism such as reproduction, variation and selection [28]. In evolutionary algorithms, candidate solutions to a problem take the role of individuals in a population, where they are evaluated in an iterative process where the set of candidate solutions evolves, similarly to the evolution of the population of a species in the natural world. Listing 2.1 shows the pseudocode for implementing an evolutionary algorithm. Popular multi-objective evolutionary algorithms include NSGA-II [34], SPEA2 [177], SMS-EMOA [10] or MOEA/D [169].

The steps of a generic evolutionary algorithm can be designed as workflow of components, as depicted in Figure 2.3. This way, a particular algorithm can be obtained by selecting particular individual components of each type. In the case of multi-objective evolutionary algorithms, replacement components typically include ranking strategies and density estimators, and the choice of using

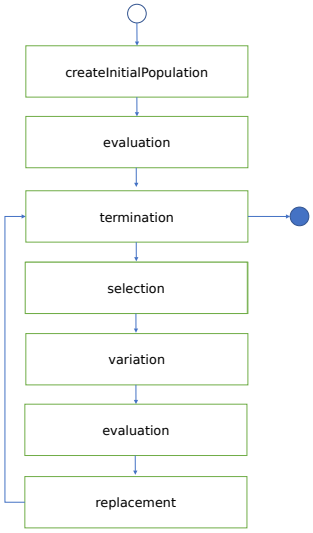


Figure 2.3: Workflow representing an evolutionary algorithm.

external archives (i.e., external populations) can be easily adopted by defining an evaluation component that stores in the archive any evaluated solution.

To facilitate the use of metaheuristics, optimization frameworks provide a set of already implemented algorithms and interfaces to facilitate the implementation and use of metaheuristics.

Definition 11: Optimization framework

An optimization framework is a software package that offers a compatible set of state-of-the-art algorithms, benchmark problems, utilities (such as quality indicators) and algorithmic templates and interfaces to easily integrate new metaheuristics or algorithms to solve optimization problems.

Popular optimization frameworks include jMetal [41] which is implemented in Java, jMetalPy [9] in Python, PlatEMO [148] in MATLAB, pymoo [15] in Python, and Pagmo [14], which is written in C++. Additionally, Pagmo has a Python wrapper called Pygmo [1]. In this PhD thesis, jMetal is the optimization framework used.

jMetal is an open-source framework implemented in Java primarily designed for multi-objective optimization tasks in the field of evolutionary and genetic algorithms. Originally, developed on 2015, it has been continuosly updated and is

[1]Pygmo: `https://github.com/esa/pygmo2`

currently in version 6 [113, 116]. jMetal aims to provide a flexible and extensible platform for researchers and practitioners to experiment with various metaheuristic algorithms, offering a wide range of functionalities, making it suitable for both educational purposes and real-world applications.

At its core, jMetal provides implementations of various evolutionary algorithms such as genetic algorithms, differential evolution, evolutionary strategies, and genetic programming, among others. These algorithms are geared towards exploring and exploiting solution spaces efficiently, with a focus on finding solutions that represent trade-offs among multiple objectives. Additionally, jMetal supports the incorporation of custom problem definitions, allowing users to model and solve their specific optimization problems effectively. The modular architecture of jMetal enables easy integration of new algorithms, operators, and problem types.

While experimenting with metaheuristics, it is important to take into account they are stochastic methods, which means that two experiments can not be directly compared using a single execution. As such, statistical tests are used to decide whether the data sufficiently support a particular hypothesis.

Definition 12: Statistical tests

A statistical hypothesis test is a statistical method used to decide whether a set of data sufficiently supports a specific hypothesis [82]. To validate whether the data supports the null hypothesis, a *p*-value computed from the test statistic and the null hypothesis is rejected if the *p*-value is less than or equal to a predefined threshold value α, which is referred to as the alpha level or significance level.

The choice of statistical test to use depends on the null hypothesis to validate and the distribution of the data [130]. The statistical tests used in this thesis are the Shapiro-Wilk and the Wilcoxon rank sum test. The Shapiro-Wilk test is a statistical test is whose null-hypothesis is that the data follows a normal distribution [136].

The Wilcoxon rank sum test (also known as Mann-Whitney U test) is a non-parametric test of the null hypothesis that, for randomly selected values X and Y from two populations, the probability of X being greater than Y is equal to the probability of Y being greater than X [105].

2.3 Quality-Diversity optimization

Another branch of optimization is Quality-Diversity optimization. This section describes the main concepts related to it, while the main use of Quality-Diversity

in this thesis is available in Section 5.9.

Definition 13: Quality-Diversity (QD) optimization

Quality-Diversity (QD) optimization is a branch of mono-objective stochastic optimization were instead of searching for the optimum of a fitness function, it searches for a large set of high-performing solutions that differ according to a few user-defined features of interest [25].

In multi-objective optimization, the algorithms search the variable space for feasible solutions that, when evaluated, are Pareto optimal, such as no other solution in the objective space improves any of the objectives without worsening at least one other objective, as every objective is considered to have the same importance. However, in Quality-Diversity there is a clear ranking of the solutions according to their evaluation according to a single objective, but the optimization process also explores a new behavior space for diversity according to a set of user-defined features. The use of multi-objective Quality-Diversity algorithms is an active area of research [18].

As a new search space for the solution's behavior is added to the problem, fitness functions must also evaluate the behavior characteristics of an individual.

Definition 14: Behavioral characteristic

In Quality-Diversity, the objective function must returns both: the fitness value, f_θ, and either a behavioral characteristic (BC) [2] or a vector of them, b_θ:

$$f_\theta, b_\theta \leftarrow f(\theta) \tag{2.17}$$

The behavioral characteristics, b_θ, describe *how* the solutions solve the specified problem, while the fitness value, f_θ quantifies *how well* it solves it. For example, if the problem to be optimized is the distance to a target on the movement of a robot, the behavioral characteristics could be the trajectory of the movement.

As the outcome of a Quality-Diversity algorithm are a set of solutions that are diverse according to the behavioral space. This set is usually called a collection, map or archive of solutions.

Definition 15: Diversity archive

The archive of solutions is a data structure were a Quality-Diversity optimization algorithm will store different "solution types" or "species". A solution type

[2] Also known in the literature as behavioral descriptor (BD) or behavioral trait (BT)

is a set of solutions to the optimization problem that show the similar behavioral characteristics. Inside each solution types, each solution competes against the rest to be maintained in the collection, as the elite individual for a specific solution type.

The simplest implementation of an archive is a n-dimensional grid, were each behavioral characteristic is discretized in fix or variable size buckets, where each bucket holds the elite of a specific solution type. One hyper-parameter when creating the archive is the "resolution" or grid-size that controls the tolerance to determine when the behavior of two individuals is similar enough to be part of the same solution type. However, there are methods to avoid the discretization of the behavior space such as distance thresholds or local density estimates, such as the average distance of the k-nearest neighbors.

The goal of a Quality-Diversity algorithm, such as MAP-Elites, is to "illuminate" or explore the whole archive with the best possible solution found for each cell.

Definition 16: MAP-Elites algorithm

The Multidimensional Archive of Phenotypic Elites, commonly known as the MAP-Elites, algorithm [109] is often used as reference of Quality-Diversity algorithms due to its simplicity. A pseudo-code of the algorithm is available at Listing 2.2. MAP-Elites is inspired by evolutionary algorithms, altering known solutions with mutation and crossover operators. These offspring solutions are evaluated and, if they improve the previous elite on a specific behavior cell, are added to the archive. Otherwise, only the best solutions is kept for each cell.

Other relevant algorithms include Novelty Search with Local Competition (NSLC) and variants of MAP-Elites such as Covariance Matrix Adaptation MAP-Elites (CMA-ME) [49] or Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) [48] that combines the self-adaptation techniques of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [62] algorithm with MAP-Elites for maintaining diversity.

2.4 Large Language Models

Since the release of GPT-3 by OpenAI, large language models have been a hot topic in both research and industry. Chapter 6 focuses on the use of large language models in the context of this doctoral thesis. This chapter defines and the main concepts related to this sub-field of natural language processing are

```
procedure add_to_archive(θ, A)
    (f_θ, b_θ) ← f(θ)
    c ← get_cell_index(b_θ)
    if A(c) = null or A(c).f < f_θ then
        A(c) ← (f_θ, θ)
    end if
end procedure

# G indicates the number of initial solutions
# I indicates the evaluation budget
procedure MAP-Elites([n_1, ..., n_d])
    A ← create_empty_archive([n_1, ..., n_d])
    for i = 1 → G do
        θ ←random_solution()
        add_to_archive(θ, A)
    end for
    for i = 1 → I do
        θ ←selection(A)
        θ' ←variation(θ)
        add_to_archive(θ', A)
    end for
    return A
end procedure
```

Listing 2.2: Pseudo-code of the MAP-Elites algorithm.

explained in this section.

Definition 17: Natural Language Processing

Natural Language Processing (NLP) is a subset of artifical intelligence that aims to allow computers to recognize, understand and generate text and speech. A recent breakthrough in NLP has been Large Language Models, that have revolutionized the field by their remarkable capabilities, including generating coherent and contextually relevant text, answering questions, summarizing information and translating languages. First, let's step back and build the way there, starting by the concept of language modeling.

Definition 18: Language modeling

Language modeling consists in the distribution estimation from a set of points $(x_1, ..., x_n)$[3], i.e., the training data, each of which is composed of a variable sequence of symbols $(s_1, ..., x_k)$[4] that must appear on a specific order and where repetitions are allowed [129]. The different allowed symbols are language-dependent and are also commonly referred to as tokens. Their ordering allows to frame language modeling as the estimation of a product of conditional probabilities, as seen in Equation 2.18.

[3]Examples of points in the training data might be sentences, paragraph or documents.
[4]Examples of symbols are characters, words, or sub-words.

$$P(x) = \prod_{i=1}^{n} P(s_n \mid s_{n-1}, s_{n-2}, \ldots s_0) \tag{2.18}$$

Definition 19: Language model

A language model (LM) is a mathematical function that provides an estimation of $P(x)$, as well as for any of the conditionals $P(s_n \mid s_{n-1}, s_{n-2}, \ldots s_0)$. Given an initial set of symbols, an LM allows to determine the probability for each existing symbol to follow that sequence, thus allowing to generate new sequences following the same pattern in training data in an auto-regressive fashion: one token at a time, and the generated token is included in the input before generating the next one.

Definition 20: Transformer architecture

Many of the state-of-the-art LMs today are based on the self-attention mechanism of the so-called Transformer architecture [155], which is pictured in Figure 2.4. Transformer is a deep-learning architecture that does not use recurrent units such as previous recurrent neural architectures, such as long short-term memory (LSTM), which were prevalently adopted for training large language models.

The key innovation of the Transformer architecture is the use of self-attention mechanisms, which allow the model to weigh the importance of symbols in an input sequence when generating an output sequence. This architecture consists of an encoder and a decoder, both of which are composed of multiple identical layers containing self-attention and feed-forward neural networks.

Definition 21: Self-attention

The self-attention mechanism, also known as scaled dot-product attention, allows a language model to consider different symbols in the context of others. For each word, it computes a score (attention score) that signifies the importance of other words when encoding a particular word. These scores are used to weight the influence of words in the encoding of a given word. This mechanism enables the model to capture various levels of dependencies in a sentence, making it particularly effective for many NLP tasks. The self-attention mechanism is what gives the Transformer its ability to handle better understand the context of a symbol and other long-range dependencies in text. This is just a high-level overview of the Transformer architecture and self-attention mechanism, for a more technical description refer to the "Attention Is All You Need" paper [155].

When it comes to computational performance, the self-attention mechanism

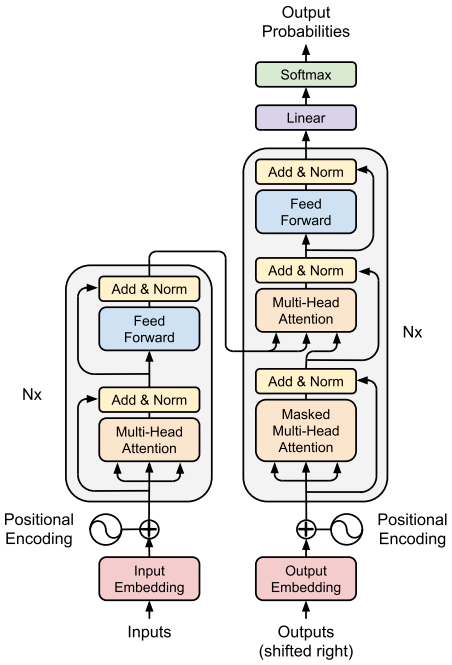


Figure 2.4: The Transformer model architecture [155].

presents both advantages and challenges compared to other architectures. On the positive side, self-attention is highly parallelizable, which means it can take full advantage of modern hardware accelerators such as GPUs. This is because the attention scores for all word pairs can be computed simultaneously, unlike recurrent architectures like Long-short-term memory where computations are inherently sequential. On the other hand, the computation of attention scores for all pairs of words leads to a quadratic increase in computational complexity with the length of the input sequence.

Definition 22: Scaling laws for neural language models

The performance of this kind of models depends on the size of both the model (i.e., the number of parameters that need to be adjusted) and the training data set [83]. Empirically, it has been shown that larger LMs are also more sampling efficient than their smaller counterparts [83]. The latter means that they are able to provide a reasonable good approximation to $P(x)$ in less training steps. This behavior has been modeled as a series of scaling laws that usually have as variable the number of parameters of the model, the dataset size, the computing cost and the value of the loss function [66, 83, 69]. As a consequence, the trend

in the last years has been to train LMs with billions of parameters.

Definition 23: Large language model (LLM)

These billion-size models are popularly referred to as Large Language Models (LLMs). For example, the OpenAI GPT3 [22], that initially served as the language model engine of the popular ChatGPT[5], featured 175 billion parameters, and the Nvidia Megatron-Turing NLG features 530 Billions of parameters [141]. Unfortunately, scaling the sizes of LLMs also comes with higher requirements of computation [83], memory [138], and energy consumption [133]. Training or performing inference with a trained LLM is normally done in supercomputers.

LLMs are usually trained in two steps [128]. The reasons for following this two-stage procedure are twofold: the high costs of training large models [83] and the lack of abundant examples specific to the use case at hand [128] (that would allow training an LLM from scratch only on that data). These two steps are referred as pre-training and adaptation.

Definition 24: Pre-training

The pre-training of an LLM is generic and requires adjusting all the model parameters for learning the structure and patterns in a language (i.e., learning $P(x)$ and the conditionals). The resulting model in this stage is often labeled as a pre-trained LLM or foundational model. Pre-trained LLMs are often made available by different companies and research institutions, lowering the requirements to benefit from LLMs, as they only need the adaptation step for the downstream task.

Some of the state-of-the-art foundational models with open licenses are the Llama family (Llama [151], Llama 2 [152] and Llama 3, each available on different number of parameters ranging from 7 to 400 billion) or Mistral, which offers an open model with 7 billion parameters [80] or models with 8x7B and 8x22B using a sparse Mixture-of-Experts architecture [81].

Definition 25: Adaptation

The second step, called adaptation, is less computationally intensive and involves adapting the LLM for a specific downstream task. This latter involves changing the conditional probabilities to generate the next symbol according to a specific use case instead of the token that would correspond in the learned language in the pre-training stage.

There are mostly two approaches for the adaptation step: zero- and few-shot

[5]https://chat.openai.com/

learning and fine-tuning.

Definition 26: Zero- and few-shot learning

Zero- and few-shot learning [22] are based on the empirical observation that, while trained to learn to predict the next symbol in a sequence, LLMs are embedding some knowledge regarding the considered training data despite not being specifically trained for that. This knowledge can be extracted from the model by providing the right input sequence of symbols. Designing such sequences is commonly referred to as prompt engineering. Zero- and few-shot learning require then no changes to the pre-trained model. Unfortunately, their success is correlated with the model size and therefore renders irrelevant when using smaller language models.

The main difference is that zero-shot learning uses prompt engineering techniques to guide the model to the correct output, while few-shot learning uses examples along the prompt to showcase example pairs of input-output to the model.

Definition 27: Fine-tuning

Fine-tuning [128] consists in further adjusting the weights of the pre-trained model, focusing on improving its performance for the downstream task. This technique is a specific method of transfer learning [172], where the idea is to transfer the learned patterns and language structure to the new task. During fine-tuning, the task-specific inputs are used to effectively transfer (i.e., modifying some parameters on the LLM while keeping the others frozen) the knowledge of a generic model into a more specialized one for a different task. Therefore, a requirement to apply fine-tuning is to have a sufficiently large set of inputs related to the downstream task. The specific fine-tuning method utilized in this PhD thesis is Low-Rank Adaptation (LoRA).

Definition 28: Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) [72] proposes freezing the foundational model weights and introduces trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters. LoRA enhances training efficiency and reduces hardware requirements by optimizing smaller matrices, while their linear design facilitates the integration of the adapter matrices with the frozen weights during deployment, ensuring no inference latency compared to fully fine-tuned models.

2.5 Semantic Web technologies

The Semantic Web is key to the contributions of this PhD thesis, as it provides the formal backbone for the proposed recommendation system. This section introduces the main concepts and Semantic Web technologies utilized through this work.

Definition 29: Semantic Web

The Semantic Web is an extension to the original Web, in which information is well defined, in a format that can be utilized not only by humans, but also by other tools or applications [67].

Semantic Web technologies are a set of open standards and tools with the goal to facilitate integration of data from different sources. These tools achieve this by easing the creation, publication and linking of data on the Web, allowing machines to process, interpret and utilize them. The most relevant ones are described in the following definitions.

Definition 30: Ontology

An ontology is a formal representation of a specific domain, simplified in a way that can be represented for a specific purpose in terms of concepts, properties and relationships [58]. Plainly, an ontology is a representation of a domain area, by defining its set of interrelated terms, properties how they are related to other terms, often via relational or logical expressions.

There are many languages to define ontologies, like OBO [140] or OWL, which will be used on this thesis and it is describe as follows.

Definition 31: Web Ontology Language (OWL)

The Web Ontology Language (OWL) [31] is a markup language, defined as a standard by the World Wide Web Consortium (W3C), used to define and publish ontologies. OWL provides a set of primitives that can be used to model the knowledge around a specific domain. These primitives take the form of axioms, statements that formally and precisely describe a specific domain, and they are classes (or concepts), properties (or attributes), instances (or class members) and relationships.

This thesis utilizes OWL-DL (Web Ontology Language - Description Logic) sublanguage, based on Description logic [5]. Table 2.1 includes a summary of the OWL-DL and Manchester syntax [68]. OWL is built on top of RDF, a graph-based markup language, but not all RDF features are included in all OWL variants.

Descriptions	Abstract Syntax	DL Syntax	Manchester Syntax
Operators	$intersection(C_1, C_2, \cdots, C_n)$ $union(C_1, C_2, \cdots, C_n)$	$C_1 \sqcap C_2 \sqcap \cdots C_n$ $C_1 \sqcup C_2 \sqcup \cdots C_n$	C_1 and C_2 and $\cdots C_n$ C_1 or C_2 or $\cdots C_n$
Restrictions	for at least 1 value V from C for all values V from C R is Symmetric	$\exists V.C$ $\forall V.C$ $R \equiv R^-$	V some C V only C R Characteristics: Symmetric
Class Axioms	$A\ partial(C_1, C_2, \cdots, C_n)$ $A\ complete(C_1, C_2, \cdots, C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots C_n$ $A \equiv C_1 \sqcap C_2 \sqcap \cdots C_n$	A EquivalentTo: C_1 and C_2 and $\cdots C_n$ A SubClassOf: C_1 and C_2 and $\cdots C_n$

Table 2.1: Comparison between the most common terms from the OWL-DL semantic syntax and the Manchester syntax. A complete list can be found at [68].

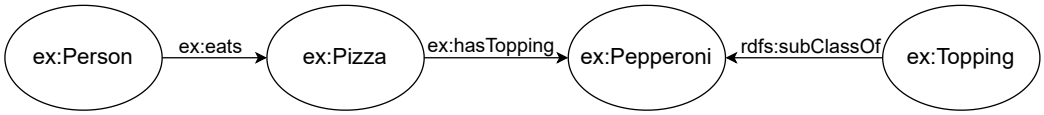


Figure 2.5: Example RDF graph.

Definition 32: Resource Description Framework (RDF)

Resource Description Framework, defined as a W3C standard, is a graph-based markup language providing a standard model for representing and exchanging data on the Web [135]. It is encouraged by the W3C in applications where data is processed or consumed by other machines or applications and not only end-users.

RDF uses unique URIs (Uniform Resource Identifier) to identify each resource on the Web. A resource is defined as a statement, represented as triples which contain a subject, a predicate and an object [145]. RDF Schema (RDFS) is an extension of RDF, allowing the definition of classes and properties of resources [145].

RDF graphs can be serialized to various syntax formats to be processed in a structured and machine-readable manner. Some of the more common serializations formats are RDF/XML, Turtle, N-Triples and JSON-LD (JSON for Linked Data). Each serialization has its own syntax rules and strengths, catering to different use cases and preferences within the realm of linked data. An example on several different serializations can be found at Listing 2.3, while Figure 2.5 shows its graph representation.

After data has been stored in RDF format, SPARQL provides an standard language for querying the graph.

```
# Example in Turtle format
@prefix ex: http://example.org/ .
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema# .

ex:Pizza ex:hasTopping ex:Pepperoni .
ex:Person ex:eats ex:Pizza .
ex:Pepperoni rdfs:subClassOf ex:Topping .
```

```
# Example in N-Triples format
<http://example.org/Pizza> <http://example.org/hasTopping>
  <http://example.org/Pepperoni> .
<http://example.org/Person> <http://example.org/eats> <http://example.org/Pizza> .
<http://example.org/Pepperoni> <http://www.w3.org/2000/01/rdf-schema#subClassOf>
    <http://example.org/Topping> .
```

Listing 2.3: Example serializations of RDF in Turtle (top) and N-Triples (bottom) format.

```
PREFIX ex: <http://example.org/>
SELECT DISTINCT ?person
WHERE {
    ?person ex:eats ?pizza .
    ?pizza ex:hasTopping ex:Pepperoni .
}
```

Listing 2.4: SPARQL query that returns all distinct person who eat Pizza with Pepperoni as topings.

Definition 33: SPARQL Protocol And RDF Query Language (SPARQL)

The SPARQL Query Language (SPARQL) [64] is the W3C standard for a query language for RDF graphs, allowing the extraction and manipulation of information over web resources identified via URIs. SPARQL queries use graph-matching to retrieve the set of RDF triples that match a pattern [123], utilizing "?" as the prefix to identify variables.

Listing 2.4 provides an example query over the example graph provided in Listing 2.3

Definition 34: Semantic Web Rule Language (SWRL)

The Semantic Web Rule Language defines new ways to describe semantic relationships between individuals in ontologies defined in OWL, adding extra inference capabilities [70]. SWRL allows the definition of rules in the form of "Antecedent $\Rightarrow$ Consequent" to represent semantic relationships. The antecedent and the consequent can be formulated as conjunctions of elements associated with one or more attributes defined by a question mark and a variable name (e.g., $?x$) in the rule. An example rule about the ongoing example can be seen in the Listing 2.5

```
ex:Pepperoni(?topping)
^ ex:hasTopping(?x, ?topping)
-> ex:Pizza(?x)
```

Listing 2.5: A SWRL rule that classifies everything that has Pepperoni has a toping as a Pizza.

After defining all the relevant technologies, lets shift the focus towards some of the main benefits obtained by implementing them.

Definition 35: Semantic reasoning

OWL ontologies use a representation based on Description Logic, a family of knowledge representation languages that enable the creation of a structured and formal definition of knowledge within a specific domain of application. Description Logics are decidable fragments of first-order logic, enabling semantic reasoning over complex logical axioms.

Reasoning is the process of utilizing algorithms to infer new knowledge from existing data by, for example, discover new connections between data not explicitly present in the original dataset [71], either via the formal specification in description logic that exists in OWL or using rules, for example in SWRL format. Ontologies are used to represent the relationships between classes, while also defining the constraints and logical rules they must satisfy. Some examples of reasoners include Pellet [139] or HermiT [137].

Definition 36: Knowledge graph

A knowledge graph is way to structure information as an interconnected network of entities and relationships as nodes and edges in a graph topology [44].

In the context of the Semantic Web, knowledge graphs are often associated with linked open data projects. Ontologies are often used in conjuction with the knowledge graph. Ontologies model the concepts and relationships, creating a shared vocabulary for a specific domain called terminological box (TBox), while the assertional box (ABox) is build as a knowledge graph, focusing on representing the specific data, following the semantic model defined by the ontology.

Chapter 3

A Semantic Approach to Standardizing Multi-Objective Optimization

This chapter proposes a semantic framework, *moody*, to consolidate multi-objective optimization knowledge in the form of an OWL ontology. *moody* follows the FAIR principles and it is validated by four use cases in the context of automatic configuration of multi-objective optimization with metaheuristics. *moody* is used to structure the knowledge graph, incorporating all data generated during this PhD thesis.

3.1 Introduction

The main challenge faced in this PhD thesis derives from the lack of a unified and standardized approach in the design of multi-objective optimization algorithms. Initially, there is no standard set of components to design these algorithms from, so each researcher working on the topic usually implements their own set of components, often similar to others previously defined. This absence hinders the comparison of results across different studies due to the lack of a unified framework for integrating diverse data. Furthermore, the process of auto-configuration [12] demands significant computational resources, as it involves generating and evaluating thousands of configurations. Identifying an existing configuration for a similar problem could save considerable time, but the methods for defining problem properties and associating them with specific configurations remain unclear.

In this context, semantic technologies have demonstrated their efficacy in various fields for integrating and representing domain knowledge, supporting data standardization, and facilitating semantic integration from multiple sources [134]. Ontologies are predominantly utilized to describe knowledge, offering a formal, logic-based approach for defining concepts and establishing a common vocabulary within a specific domain [145].

Addressing these needs, this chapter introduces a semantic framework, guided by the FAIR principles (Findable, Accessible, Interoperable, and Reusable) [157], to consolidate knowledge in multi-objective optimization using an OWL ontology. This framework also enables semantic reasoning in analyzing algorithm configurations or optimization experiments [71].

The main contributions are as follow:

- The development of an ontology called *moody* (Multi-Objective Optimization ontologY)[1], which formalizes aspects of multi-objective evolutionary algorithms, their parameters, continuous multi-objective problems, the landscapes of their search spaces, and the required quality indicators for assessing algorithm performance. The ontology formalizes the set of algorithm parameters for well-known evolutionary algorithms like NSGA-II or MOEA/D, which helps to integrate configurations of different algorithm implementations into a knowledge graph, and export them, if they are compatible, to a different implementation of the algorithm.
- The creation of a knowledge graph, populated with algorithm configurations and optimization experiments, annotated semantically as per the de-

[1] Available on permanent URL: `https://w3id.org/moody`

fined ontology and formatted in RDF[2]. This semantic framework empowers advanced reasoning capabilities over the knowledge graph, showcasing its efficacy in validating configurations of multi-objective algorithms within the domain of algorithm auto-configuration. Moreover, the knowledge graph serves as a foundation to provide users recommendations [161] of superior configurations of algorithms beyond their default settings, as describe in following chapters.

- The implementation of four use cases to validate the semantic approach in the context of automatic configuration of multi-objective evolutionary algorithms. These include a detailed exploration of how *moody* can enhance an auto-configuration tool, the integration of algorithm configuration and experiments from varied sources for new experiment validation, SPARQL queries for extracting insights from the knowledge graph, and the integration and exporting this knowledge into optimization frameworks used in real-world applications, like pagmo [14] from the European Space Agency.

The rest of this chapter is structured as follows. In Section 3.2, a literature overview is given. Section 3.3 gives an in-depth description of the semantic approach followed, focusing on the ontology model. Four use cases to validate this approach are defined in Section 3.4, and are later discussed in Section 3.5.

3.2 Related works

The primary objective of the proposed ontology is to formalize algorithms, parameters, problems, and quality indicators within the multi-objective optimization domain. This formalization aims to facilitate the creation of a knowledge graph comprising algorithm configurations, capable of assimilating experiments from diverse and heterogeneous sources, empowering semantic reasoning over this knowledge graph.

Several studies have employed ontologies and metaheuristics. [86] defines an ontology for annotating performance tests among continuous optimization algorithms using BBOB (Black-Box Optimization Benchmarking) tests. [158] uses ontologies to incorporate terminology and formal definitions with the developer's reasoning and justifications for the optimization model about the ideas and assumptions of each model. However, none of them addresses the use of ontologies for the formal description of configurations and problems, aiming to find the best possible configuration for an algorithm or the best algorithm to solve a problem.

[2]The knowledge graph can be accessed at `https://doi.org/10.5281/zenodo.7458095`

In the field of multi-objective optimization, the use of ontologies to formalize knowledge has not yet been widely accepted, with only a few cases noted. After reviewing the available literature, the only ontology on multi-objective optimization is PMOEA ontology (Preference-based Multi-Objective Evolutionary Algorithm) [3] [90, 91]. PMOEA focuses mainly on preference-based algorithms, preference models and preference integration to guide the algorithms. This focus on a small subset of the evolutionary algorithms field makes this ontology not broad enough for the proposed framework.

There are other published ontologies with a more focused scope inside multi-objective optimization. Examples of these ontologies are for modeling constraints in multi-objective optimization algorithms [7] or the domain knowledge of the field of the multi-objective problem [127, 99, 24]. These ontologies are transversal to *moody*. *moody* focuses on the formalization of the algorithms themselves, while the ontologies mentioned here attempt to use domain knowledge to help solve a problem in a specific domain.

One ontology with similar objectives of *moody* is OPTION (OPTImization algorithm benchmarking ONtology) [86], which is aimed at making benchmarking more reusable and interoperable, but it is centered only on single-objective optimization. OPTION and *moody* are complementary, each focusing on a different part of the optimization field, so mappings have been established between them (see next section).

3.3 Semantic approach

moody is implemented as an OWL 2 ontology following the FAIR principles and designed following the *Ontology Development 101* methodology [118] as follows:

1. **Determine the domain and scope of the ontology**. The goal of *moody* is to be able to model experimentation in multi-objective optimization problems and algorithms. It provides a framework that allows algorithms to be formalized with all their configurable parameters and problems defined by their defining characteristics. The scope of *moody* is focused on the most relevant algorithms (NSGA-II [34], MOEA/D [169] and MOEA/D-DE [89]) and benchmark problems (Deb-Thiele-Laumanns-Zitzler (DTLZ) [36], Gu-Liu-Tan (GLT) [59], Li-Zhang 2009 (LZ09) [89], REal-world problems (RE) [147], CEC 2009 competition (UF) [170], Zitzler-Deb-Thiele (ZDT) [175] and Walking-Fish-Group (WFG) [76] problem families) to highlight the

[3] Later renamed PMOMH (Preference-Based Multiobjective Metaheuristics)

features of the ontology, but *moody* is designed to be easily extended with new algorithms and real-world problems.

2. **Consider reusing existing ontologies**. To align the proposed ontology with existing vocabularies, several ontologies have been reused. Firstly, the Data Mining Optimization Ontology (DMOP) [84] has been used to describe the parameters for the optimization algorithms. From the big data analytics OWL 2 ontology (BIGOWL) [6], the definition of the concepts of algorithm and problem is reused. The OPTImization algorithm benchmarking ONtology (OPTION) [86] is modeling single-objective optimization, but some terms are common on both domains, so mappings have been created between them, such as *moody:QualityIndicator* that maps to *ontoopt:performance_evaluation_function* or *moody:ProblemResolution* and *ontoopt:algorithm_execution*.

3. **Enumerate important terms in the ontology**. The main terms of *moody* are the *Experiment*, the *Algorithm*, the *Parameter*, the *Problem*, the *Quality Indicator* and the *Problem Resolution*, which includes the results of each execution in the *Experiment*.

4. **Define classes and the class hierarchy**. The classes of the ontology model are the key concepts defined in the previous point and more specific classes to model concrete *Algorithms* or *Problems*. Figure 3.1 shows all the classes defining the key concepts. For example, *Algorithm* is a concept defined at BIGOWL, but *moody* expands it with the subclasses *NSGA-II* and *MOEA/D*. These high level classes allow the ontological model to be easily expanded when a use case needs new, more specific algorithms. They can be easily integrated as new subclasses of *Algorithm*, inheriting all the semantic relationships from it.

5. **Define the properties of classes and slots** [4]. Object properties are used to define relationships between classes. For example, an *Experiment* is *evaluated by* a *Quality Indicator* or a *Parameter* is *compatible with* an *Algorithm*. Data properties are used to define what attributes an instance of a class has. One example of data properties is *Algorithm* which has a *crossover* type, a *population size*, between others.

6. **Define the facets** [5] **of the slots**. All properties in the ontology are constrained by their range and domain; for example, the object property *evaluated by* has a range of *Experiment* and a domain of *Quality Indicator*. For the data properties, the domain is specified at the parent property, while

[4] On Ontology 101, slots refers to properties of classes and instances
[5] On Ontology 101, facets refers to constraints of properties

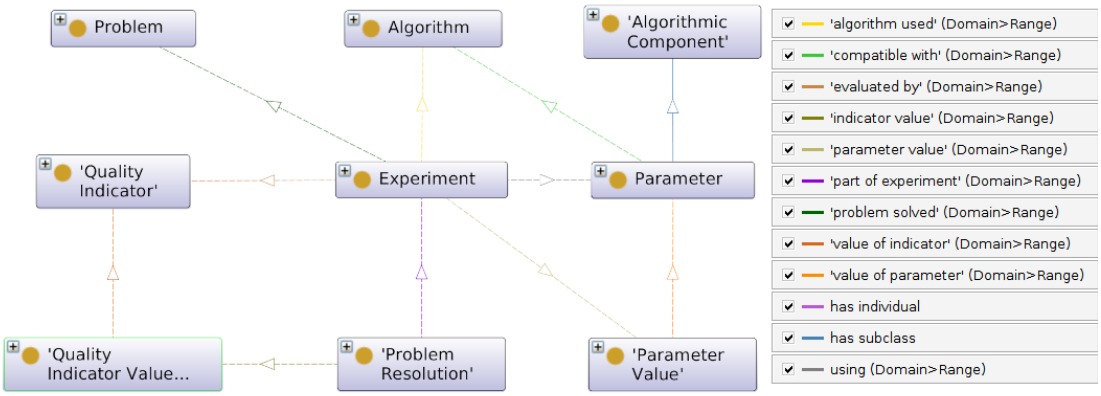


Figure 3.1: Class diagram of *moody*. Continuous arrows refer to subclass of. Dotted arrows refer to specific properties as shown in the figure legend at right.

the range is specific to each property. Data properties range is not defined only by its type but also by the possible values. These range definitions allow a semantic reasoner, like Pellet [139], to validate if an algorithm configuration is valid, as showcased in Section 3.4. Table 3.1 shows the range of all algorithm parameters.

7. **Create instances**. Individuals are specific data points that belong to a class. These instances are created by mapping experimentation data to RDF by following the model defined by the ontology. For this PhD thesis, an auto-configuration tool has been developed to generate instances of configurations and map the to RDF creating a knowledge graph; more information about it is available on Chapter 5.

3.3.1 Ontology model

The *moody* ontology has a total of 59 classes, 11 object properties, 81 data properties and 580 logical axioms. The focus of the proposed ontology is bifold; on the one hand, the ontology formally defines multi-objective optimization problems, the algorithms used to solve them and the quality indicators used to validate the quality of the solutions. On the other hand, *moody* provides a framework for the semantic annotation of multi-objective optimization experiments.

Figure 3.1 depicts the main classes of *moody* and Table 3.2 shows the formal definition in description logic of the object properties in the ontology. A short description of the most important classes is included next:

- ***Problem*** of multi-objective optimization, meant to be solved in a *Experi-*

Parameter	Value Range
Aggregative function	{"PenaltyBoundaryIntersection", "Tschebycheff", "WeightedSum"}
Algorithm result	{"externalArchive", "population"}
BLX alpha crossover alpha value	xsd:double[>= "0.0"^^xsd:double, <= "1.0"^^xsd:double]
Create initial solutions	{"latinHypercubeSampling", "random", "scatterSearch"}
Crossover probability	xsd:double[>= "0.0"^^xsd:double, <= "1.0"^^xsd:double]
Crossover repair strategy	{"bounds", "random", "round"}
Crossover	{"BLX_ALPHA", "SBX", "wholeArithmetic"}
Maximum number of evaluations	xsd:integer
Maximum number of replaced solutions	xsd:integer
Mutation probability	xsd:double[>= "0.0"^^xsd:double, <= "1.0"^^xsd:double]
Mutation repair strategy	{"bounds", "random", "round"}
Mutation	{"polynomial", "uniform"}
Neighborhood selection probability	xsd:double[>= "0.0"^^xsd:double, <= "1.0"^^xsd:double]
Neighborhood size	xsd:integer
Offspring population size	xsd:integer
Polynomial mutation distribution index	xsd:double[>= "5.0"^^xsd:double, <= "400.0"^^xsd:double]
Population size	xsd:integer
Population size with archive	xsd:integer
SBX crossover distribution index	xsd:double[>= "5.0"^^xsd:double, <= "400.0"^^xsd:double]
Selection tournament size	xsd:integer[>=2, <=10]
Selection	{"random", "tournament"}
Uniform mutation perturbation	xsd:double[>= "0.0"^^xsd:double, <= "1.0"^^xsd:double]

Table 3.1: Ranges for the possible parameters of an algorithm. The domain of all parameters is *Parameter Value*. Type of a value is xsd:string unless specified.

ment. *moody* supports characterizing each problem through a series landscape characteristics [94] obtained through a sampling of the search space. Some of the characteristics included are: the number of variables and objectives, the average and maximum distance among solutions in the variable and objective space, the proportion of non-dominated solutions or the average proportion of dominated solutions between neighbors. *moody* currently contains the following families of problems with their characteristics: DTLZ, GLT, LZ09, RE, UF, WFG and ZDT. This process of sampling the landscape characteristics is explained in more detail in Chapter 4.

- ***Algorithm*** to solve a multi-objective optimization *Problem*, and it is used in a *Experiment*. Each *Algorithm* is compatible with a series of *Parameters* that

Object Properties	Description Logic
algorithmUsed	$\exists$ algorithmUsed Thing $\sqsubseteq$ Experiment $\top \sqsubseteq \forall$ algorithmUsed Algorithm
compatibleWith	$\exists$ compatibleWith Thing $\sqsubseteq$ Parameter $\top \sqsubseteq \forall$ compatibleWith Algorithm
evaluatedBy	$\exists$ evaluatedBy Thing $\sqsubseteq$ Experiment $\top \sqsubseteq \forall$ evaluatedBy QualityIndicator
indicatorValue	$\exists$ indicatorValue Thing $\sqsubseteq$ ProblemResolution $\top \sqsubseteq \forall$ indicatorValue QualityIndicatorValue
parameterValue	$\exists$ parameterValue Thing $\sqsubseteq$ Experiment $\top \sqsubseteq \forall$ parameterValue ParameterValue
partOfExperiment	$\exists$ partOfExperiment Thing $\sqsubseteq$ ProblemResolution $\top \sqsubseteq \forall$ partOfExperiment Experiment
problemSolved	$\exists$ problemSolved Thing $\sqsubseteq$ Experiment $\top \sqsubseteq \forall$ problemSolved Problem
using	$\exists$ using Thing $\sqsubseteq$ Experiment $\top \sqsubseteq \forall$ using Parameter
valueOfIndicator	$\exists$ valueOfIndicator Thing $\sqsubseteq$ QualityIndicatorValue $\top \sqsubseteq \forall$ valueOfIndicator QualityIndicator
valueOfParameter	$\exists$ valueOfParameter Thing $\sqsubseteq$ ParameterValue $\top \sqsubseteq \forall$ valueOfParameter Parameter

Table 3.2: List of the most relevant object properties in *moody* formally defined using the description logic syntax.

modify its behavior. Widespread algorithms from the state of the art have been added to the ontology as a reference, like NSGA-II, MOEA/D and MOEA/D-DE.

- *Parameter* of an *Algorithm*. Implementations of metaheuristics allow changing the components of the *Algorithm*, such as changing the crossover function, without developing an entirely new *Algorithm*. This flexibility in configurations is particularly useful in the application of techniques of auto-configuration of algorithms to optimize a metaheuristic to a specific problem (or set of problems). In the current literature, each implementation defines its parameters, making it difficult to compare two implementations. *moody* aims to provide a standard definition of the parameters of the algorithms from the state of the art. *Parameters* are compatible with a series of *Algorithms* and are used in a *Experiment*.

- *Algorithmic Component* is a subclass of *Parameter* that models the main components of a metaheuristic. This *Algorithmic Components* are: the creation of initial solutions, termination criterion, selection, variation and replacement components. How these components form a generic evolutionary algorithm is described in Listing 2.1.

- *Quality Indicators* are used as metrics to quantify the quality of the non-

dominated front after the process of optimization by a metaheuristic. They evaluate each *Problem Resolution* in a *Experiment*. The indicators usually measure the current front's quality in terms of convergence, diversity or both. Currently, the most common quality indicators for MOPs have been included: Hypervolume [173], Spread [34], Inverted Generational Distance [29], Inverted Generational Distance Plus [78] and Additive Epsilon [85].

- Problem Resolution defines a single execution of an *Experiment*. It is the execution of a single *Algorithm*, using a defined set of *Parameters*, to solve a specific *Problem*. This execution is evaluated with one or more *Quality Indicator*, having one single *Quality Indicator Value* each.

- Experiment defines a series of independent *Problem Resolution*, utilizing one *Algorithm* with a fixed set of *Parameters* to solve a *Problem* while evaluated by a series of *Quality Indicators*.

3.3.2 Data consolidation

Once the ontology is defined, it is linked with other ontologies like DMOP, BIGOWL, and OPTION. The linked ontology constitutes the terminological box (TBox) of the proposed semantic framework. The assertional box (ABox) considers all the instances containing the data modeled. The ABox is built as a knowledge graph guided by the ontology. In order to create a solid base for the knowledge graph, an auto-configuration framework, like irace [103] or the one proposed in Chapter 5, is applied to jMetal [111] to produce configurations for the problems mentioned in the previous section, standardized to RDF format using mapping functions.

A subset of the knowledge graph is shown in Figure 3.2 as an example of a configuration graph structure. Figure 3.3 provides an overview of the semantic model[6] and how data gets ingested into the knowledge graph. Section 3.4.2 illustrates a sample case demonstrating the process of transforming an algorithmic configuration into a knowledge graph according to the *moody* ontology knowledge.

A reference implementation of a tool to ingest into RDF is described in Chapter 7 [7]. This reference implementation in Python demonstrates how to load to the knowledge graph configurations generated from the tools provided in this PhD thesis. The Python script processes each configuration from the auto-configuration tool's output. It incorporates each configuration into the knowl-

[6]The linked Open Data Cloud comes from `https://lod-cloud.net/`

[7]The implementation is available at: `https://gitlab.com/jfaldanam-phd/recommoonder`

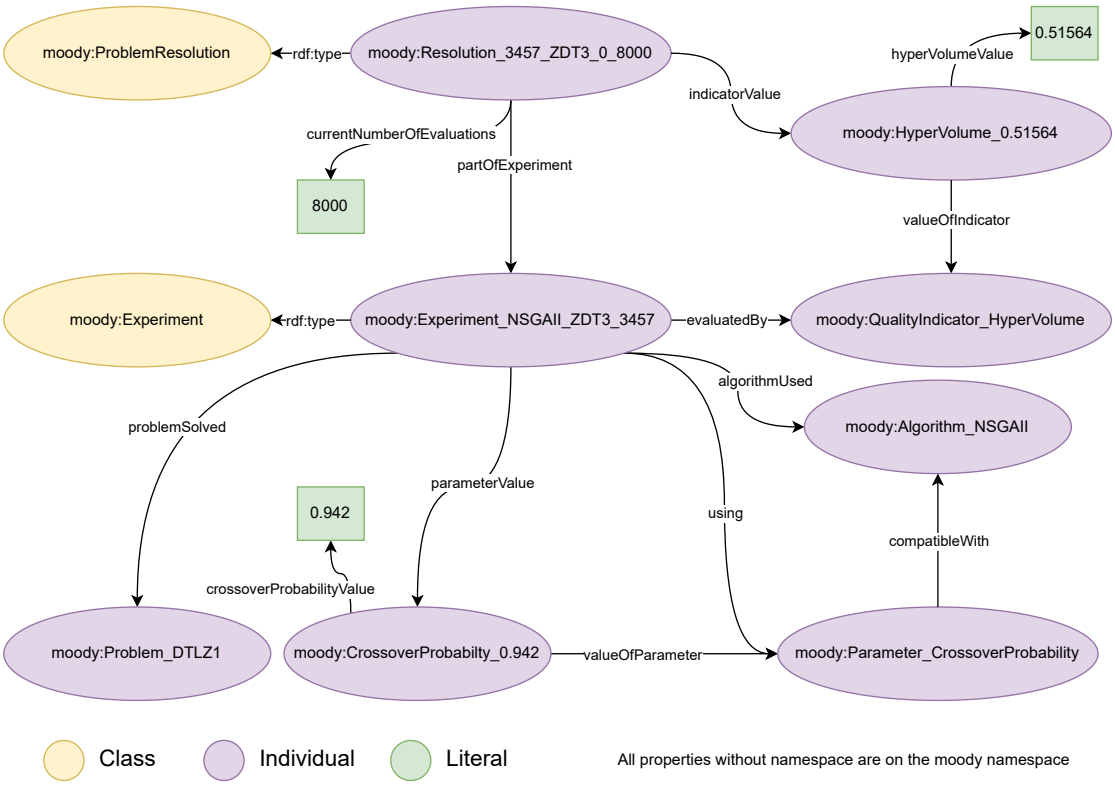


Figure 3.2: Graph representation of a the knowledge graph.

edge graph as an Experiment, specifying their properties in accordance with the structure defined by the ontology. This ensures that each configuration's details are captured and formally integrated into the broader context of the knowledge graph.

3.4 Validation

Four defined use cases validate the proposed *moody* ontology, highlighting its key features. These use cases focus tasks such as performing semantic validation of algorithm configurations, integrating existing knowledge from prior studies for new experiments, executing sample queries on the resultant knowledge graph to obtain valuable insights from the data, and, finally, exporting configurations to various implementations.

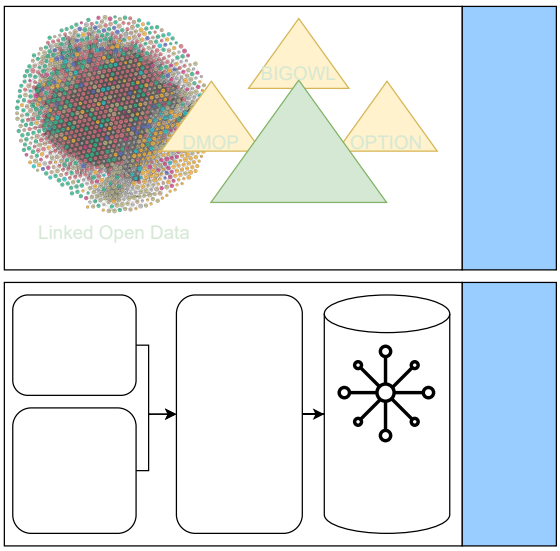


Figure 3.3: General overview of the semantic framework *moody*.

3.4.1 Supporting auto-configuration

During the auto-configuration process, tools typically generate hundreds and thousands of configurations, and some of them may lack significant differences. This can occur due to various reasons, including negligible distinctions between continuous parameters or, depending on the tool, a lack of consideration for dependencies among parameters. For instance, it may evaluate two configurations that utilize random selection but differ in the selection tournament size, even though the tournament size parameter is only relevant when the tournament component is employed as the selection method.

In this context, the utilization of a semantic model and the creation of a knowledge graph can provide interesting options for auto-configuration packages. Firstly, by leveraging the knowledge graph populated with past executions and the formal specification of parameter hierarchy, knowledge can be utilized to avoid executing configurations that are known to yield poor results. Furthermore, by utilizing knowledge of previous executions as a starting point, a semantically-enriched auto-configuration tool can enhance the initial state and facilitate more efficient configuration processes.

Secondly, having a formalized definition for the algorithm and its parameters allows a SWRL engine to execute semantic reasoning and infer new knowledge from the semantically annotated data. There are two ways to use a semantic reasoner to validate configurations: one is to validate that every property has a valid domain and range and the other is to use a reasoner to validate algorithm

```
moody:Experiment(?x)
^ moody:SbxCrossoverDistributionIndex(?p)
^ moody:sbxCrossoverDistributionIndexValue(?pv, ?cv)
^ moody:parameterValue(?x, ?pv)
^ moody:valueOfParameter(?pv, ?p)
^ moody:Crossover(?p2)
^ moody:crossoverValue(?pv2, ?cv2)
^ moody:parameterValue(?x, ?pv2)
^ moody:valueOfParameter(?pv2, ?p2)
^ swrlb:equal(?cv2, "BLX_ALPHA")
-> moody:InvalidExperiment(?x)
```

Listing 3.1: A SWRL rule to validate when parameters that depend from the SBX crossover are used on an algorithm that uses BLX_Alpha as the crossover operation.

configurations automatically. Due to the Open World Assumption, it is not possible to check directly if a configuration is valid. To validate a configuration, SWRL rules are created to assert whether a configuration is invalid.

For the validation of the domain and range of the properties, *moody* formalizes both data and object properties (examples of each are described in Tables 3.1 and 3.2, respectively). A semantic reasoner will check if those restrictions are met, and if they are not, it will point out where the inconsistency lies.

For the automatic validation of configurations, the SWRL rules defined in Listings 3.1 and 3.2 check cases where a configuration is invalid because it mixes configuration parameters of two crossover operators, SBX and BLX_Alpha. Listing 3.1 reads as follows: "Being **?x** an experiment with a parameter **?p2** of type Crossover and a parameter **?p** of type SbxCrossoverDistributionIndex if both parameters have a value, and the value of Crossover is 'BLX_ALPHA', then **?x** is an invalid configuration". This rule checks that parameters that depend on the SBX crossover are not used with the BLX_Alpha crossover, as they are incompatible.

Integrating these two approaches into auto-configuration tools would enable the acceleration of the execution process by pruning sections of the search space that have previously been explored with unfavorable outcomes or being discarded due to semantic reasoning. Moreover, it guarantees the validity of the configurations being evaluated.

3.4.2 Data integration

Semantic technologies allow data integration from several sources in a standardized format, RDF. The configurations provided in a previous study can be semantically annotated using an ontology to model the data. To demonstrate this idea, the study presented in [115] is included, where NSGA-II is auto-configured us-

```
moody:Experiment(?x)
^ moody:BlxAlphaCrossoverAlphaValue(?p)
^ moody:blxAlphaCrossoverAlphaValueValue(?pv, ?cv)
^ moody:parameterValue(?x, ?pv)
^ moody:valueOfParameter(?pv, ?p)
^ moody:Crossover(?p2)
^ moody:crossoverValue(?pv2, ?cv2)
^ moody:parameterValue(?x, ?pv2)
^ moody:valueOfParameter(?pv2, ?p2)
^ swrlb:equal(?cv2, "SBX")
-> moody:InvalidExperiment(?x)
```

Listing 3.2: A SWRL rule to validate when parameters that depend from the BLX_Alpha crossover are used on an algorithm that uses SBX as the crossover operation.

ing the WFG family of problems and later used to solve the WFG and DTLZ families against the default configuration of NSGA-II and other metaheuristics. Listing 3.3 shows the configuration of the NSGA-II found in the study and the result of the algorithm solving the DTLZ1 problem with it in RDF serialized as turtle. Figure 3.4 shows a subset of this configuration as a graph. The ingesting of this data to the knowledge graph depends on how the configurations were defined in the original source. However, custom mapping functions or R2RML [8] rules can be defined to facilitate the process.

As previously mentioned, the auto-configuration of algorithms yields a substantial volume of configurations. Employing mapping functions to assimilate these configurations into a knowledge graph, guided by the *moody* semantic framework, opens avenues for in-depth analysis, such as validation of the configurations, further reasoning analysis and the execution of SPARQL queries over them.

3.4.3 Querying the knowledge graph

Once the data for a series of experiments have been ingested into the knowledge graph guided by the ontology, the semantic model can be used to query the data using the SPARQL query language. This use case shows sample SPARQL queries that can be used to get insights into the knowledge graph. One first sample in Listing B.1 includes the SPARQL query defined to obtain the parameter configuration from a specific experiment. This query is a common yet useful query as most other analyses will only return the experiment URI.

When experimenting with new algorithm configurations to solve a particular problem, you can consult the knowledge base to find the best configurations that

[8] https://www.w3.org/TR/r2rml/

```
###  https://w3id.org/moody#Experiment_NSGAII_DTLZ1_GECCO19
moody:Experiment_NSGAII_DTLZ1_GECCO19 rdf:type owl:NamedIndividual ,
        moody:Experiment ;
    moody:algorithmUsed moody:Algorithm_NSGAII ;
    moody:evaluatedBy moody:QualityIndicator_HyperVolume ;
    moody:parameterValue moody:ParameterValue_AlgorithmResult_externalArchive ,
        moody:ParameterValue_BlxAlphaCrossoverAlphaValue_0.5906 ,
        moody:ParameterValue_CreateInitialSolutions_latinHypercubeSampling ,
        moody:ParameterValue_CrossoverProbability_0.9874 ,
        moody:ParameterValue_CrossoverRepairStrategy_bounds ,
        moody:ParameterValue_Crossover_BLX_ALPHA ,
        moody:ParameterValue_ExternalArchive_crowdingDistanceArchive ,
        moody:ParameterValue_MaximumNumberOfEvaluations_25000 ,
        moody:ParameterValue_MutationProbability_0.0015 ,
        moody:ParameterValue_MutationRepairStrategy_random ,
        moody:ParameterValue_Mutation_polynomial ,
        moody:ParameterValue_OffspringPopulationSize_200 ,
        moody:ParameterValue_PolynomialMutationDistributionIndex_158.05 ,
        moody:ParameterValue_PopulationSizeWithArchive_20 ,
        moody:ParameterValue_PopulationSize_100 ,
        moody:ParameterValue_ProblemName_dtlz.DTLZ1 ,
        moody:ParameterValue_ReferenceFrontFileName_DTLZ1.csv ,
        moody:ParameterValue_SelectionTournamentSize_9 ,
        moody:ParameterValue_Selection_tournament ,
        moody:ParameterValue_Variation_crossoverAndMutationVariation ;
    moody:problemSolved moody:Problem_DTLZ1 ;
    moody:using moody:Parameter_AlgorithmResult ,
        moody:Parameter_BlxAlphaCrossoverAlphaValue ,
        moody:Parameter_CreateInitialSolutions ,
        moody:Parameter_Crossover ,
        moody:Parameter_CrossoverProbability ,
        moody:Parameter_CrossoverRepairStrategy ,
        moody:Parameter_ExternalArchive ,
        moody:Parameter_MaximumNumberOfEvaluations ,
        moody:Parameter_Mutation ,
        moody:Parameter_MutationProbability ,
        moody:Parameter_MutationRepairStrategy ,
        moody:Parameter_OffspringPopulationSize ,
        moody:Parameter_PolynomialMutationDistributionIndex ,
        moody:Parameter_PopulationSize ,
        moody:Parameter_PopulationSizeWithArchive ,
        moody:Parameter_ProblemName ,
        moody:Parameter_ReferenceFrontFileName ,
        moody:Parameter_Selection ,
        moody:Parameter_SelectionTournamentSize ,
        moody:Parameter_Variation .

###  https://w3id.org/moody#Resolution_GECCO19_DTLZ1_0_25000
moody:Resolution_GECCO19_DTLZ1_0_25000 rdf:type owl:NamedIndividual ,
        moody:ProblemResolution ;
    moody:indicatorValue moody:QualityIndicatorValue_HyperVolume_0.00353 ;
    moody:partOfExperiment moody:Experiment_NSGAII_DTLZ1_GECCO19 ;
    moody:currentNumberOfEvaluations 25000 .
```

Listing 3.3: RDF data of a NSGA-II configuration to solve the DTLZ1 problem in turtle format.

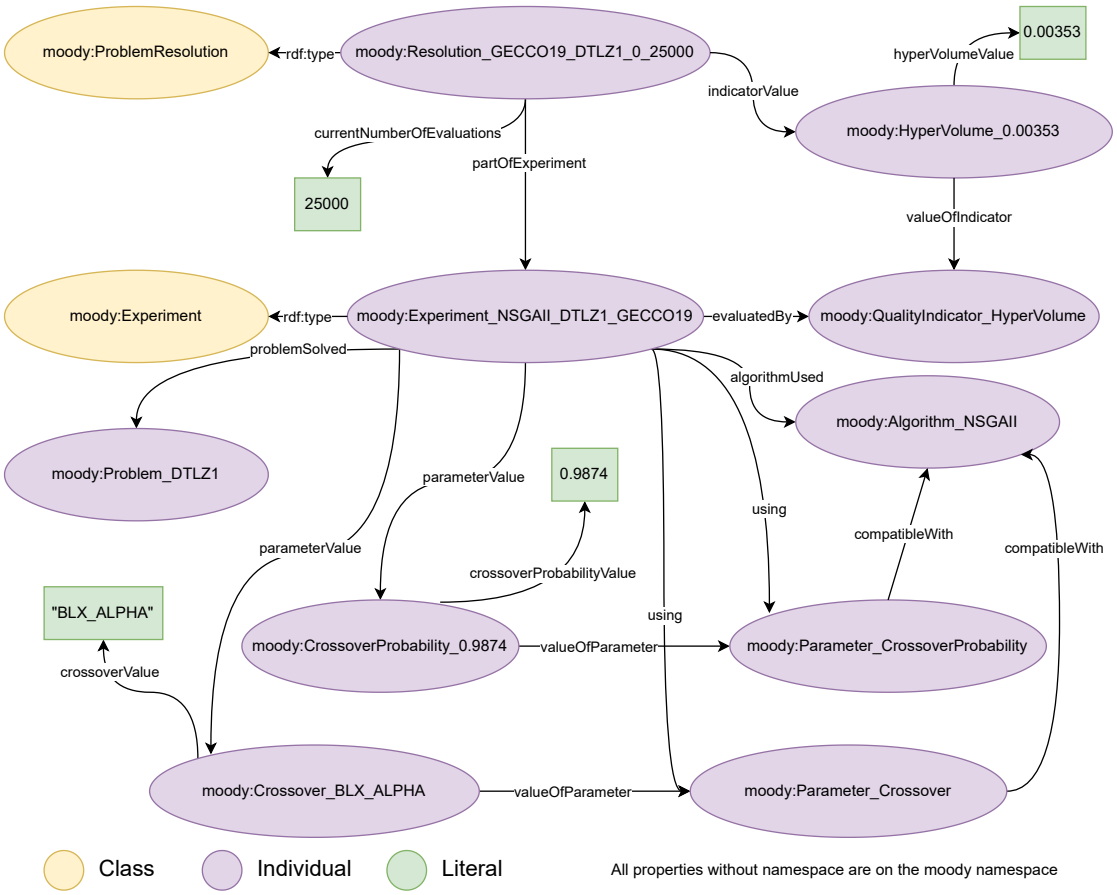


Figure 3.4: Graph representation of a subset of a NSGA-II configuration to solve the DTLZ1 problem.

have been discovered. The second query, as described in Listing B.2, provides the answer to the question of which algorithmic configuration is most effective in solving a specific problem, based on a particular quality indicator. This query takes into account that metaheuristics, in general, are not deterministic, so it calculates the average value from multiple experiment executions.

A derived query to check for the best performing algorithm for problems with similar characteristics can be seen in Listing B.3 for problems with a disconnected front. Comparing problems by their landscape helps transfer the knowledge of good configurations from one problem to a new, similar problem. Chapter 4 delves deeper on how to group problems with similar landscape.

3.4.4 Exporting configurations to different frameworks

As mentioned in Section 3.4.2, configurations from diverse sources can be integrated. However, the formal specification provided by *moody* allows to extract the best configuration that can be compatible with different implementations of an algorithm.

In this use case, a scenario is devised where a knowledge graph is populated with configurations of the NSGA-II multi-objective evolutionary algorithm obtained with different sources (such as irace and jMetal, as mentioned before) for a set optimization problems. In this context, a user is interested in using the NSGA-II algorithm included in pagmo, a framework for massively parallel optimization developed by the European Space Agency [14]; concretely, the user would like to get a configuration of NSGA-II for a particular problem (ZDT4). So, instead of trying to find that configuration using pilot tests, that is a trial-and-error process, or using an automatic configuration tool (which can take hours of computation), an alternative is to query the knowledge graph for a stored configuration of NSGA-II for that problem.

The NSGA-II implementation of pagmo only allows to set the parameters of the standard NSGA-II for continuous optimization, which are restricted to the distribution indices of the SBX crossover and Polynomial mutation and the mutation and crossover probabilities. The parameters mentioned are a subset of those being annotated in the ontology. Therefore, when querying for the result, the previous queries are extended to find configurations that utilize the values of the standard version of NSGA-II. Listing B.4 demonstrate how to query for configurations that are compatible with pagmo. Small tolerances are given to floating point values to ensure that all relevant configurations are being retrieved.

Figures 3.5 shows an example of the fronts that can be obtained with the NSGA-II included in pagmo for the considered problem. The front on the left is obtained using the standard NSGA-II settings and the front on the right is the corresponding one to run the algorithm using the configuration returned by *moody*.

3.5 Discussion

There are two main reasons for the use of semantic technologies to model the configurations of evolutionary algorithms. First, they are open standards for semantic technologies defined by the W3C for the publication and integration of data and are supported by open tools, allowing different optimization frameworks or algorithmic implementations to store or retrieve configurations from

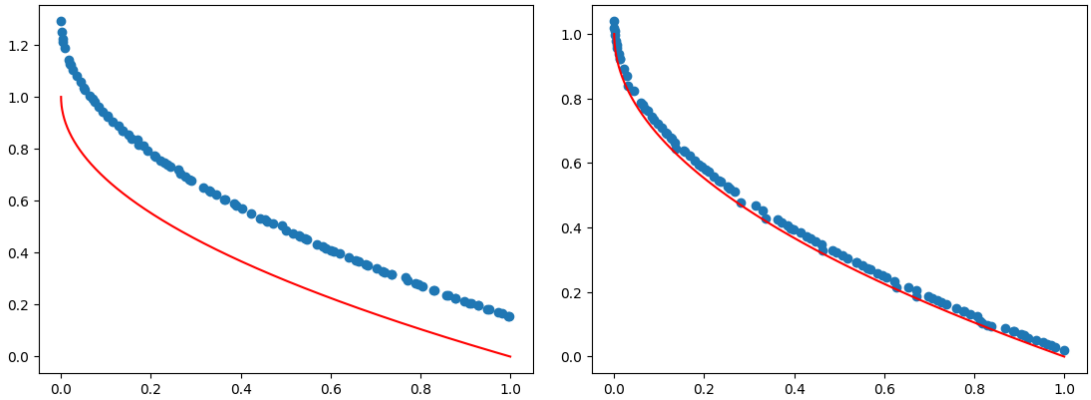


Figure 3.5: In red, reference front for the ZDT problem. In blue, Front for the ZDT4 problem obtained by NSGA-II with standard settings (left), and front obtained by exporting a configuration from *moody* (right). The target framework is pagmo.

a implementation-agnostic knowledge graph, as well as exporting them to each implementation via mapping functions. The other driving factor behind the use of semantic web technologies is to present data in a format that can be utilized not only by humans, but also by other tools or applications.

While these open standards are widely utilized, it is important to note that not all practitioners may be familiar with them. Consequently, the primary limitation of the proposed semantic framework is the requirement for users to acquaint themselves with these standards in order to integrate their applications or tools with *moody*. However, the benefits of said standards (automatic integration or open tools like reasoners) make it worthwhile trade-off for *moody*.

The field of metaheuristics sees a continuous influx of algorithm proposals, often referred as new techniques, though in many cases they are variations of metaphor-based approaches like evolutionary algorithms, particle swarm optimization, or ant colony optimization [143, 23, 168]. In this sense, the use of an ontology such as *moody* can be the basis of a formalization of metaheuristic families, including their constituting components and their relationships. This would help in detecting when a new proposal is in fact a variant of existing algorithms, which would have a positive impact in the potential users of such technique in terms of clearly understanding their principles of working. Taking as an example the workflow of evolutionary algorithms included in Figure 2.3, describing the components in a precise way would permit to determine that a metaheuristic that fits into such workflow (i.e., its main components are a kind of selection, variation and replacement strategies) can be considered as an evolutionary algorithm.

The presented use cases illustrate scenarios where this proposal demonstrates its utility applied to the process of auto-configuration of algorithms and its broader application as a comprehensive framework for integrating data that can subsequently be exported to any framework via mapping functions.

Chapter 4

Similarity Between Multi-Objective Problems Via Landscape Analysis

This chapter focuses on a critical challenge in recommending algorithmic configurations: defining a metric to measure the similarity between multi-objective problems from a set of landscape characteristics. Additionally, in this chapter *moorphology* is presented as a tool to sample from the variable and objective search space of a multi-objective problem and calculate the required landscape characteristic from said sampling. In the context of this PhD thesis, the proposed similarity metric is key in the recommendation of algorithmic configurations for solving unknown problems. At the same time, *moorphology* generated the main characteristics that are used during the recommendation process.

4.1 Introduction

In the context of optimization, landscape analysis is the field that studies the topological and structural characteristics of optimization problems. From the topology of the problem, a set of features are calculated which characterize different details about the landscape, such as ruggedness or multi-modality. These landscape features allow for deeper analysis on the behavior of metaheuristics and they provide the opportunity to select the best algorithm for new unseen problems [94]. However, it is still unclear what is the important feature(s) that makes two problems similar [122].

To implement a recommendation system, two complimentary components are required: a set of landscape features that characterizes the space sufficiently to separate the algorithms based on their performance, and a comprehensive set of benchmark problems to supply a large-enough knowledge base to train and evaluate a predictive model. While [94] defines the former, a practical implementation of said set of landscape characteristics is required. The knowledge base required by the latter is provided by the work in this PhD thesis, where Chapter 3 provides the semantic structure to create the required knowledge graph and Chapter 5 provide the tools to grow the knowledge graph to the required size.

Morphology, defined as "the branch of biology that deals with the form and structure of organisms without consideration of function"[1], have inspired the name for *moorphology* [2], a software tool that aims to fill the gap as an implementation for the analysis of the landscape features of multi-objective optimization (MOO) problems. *moorphology* is based on the jMetal implementation of multi-objective problems, using [94] as the source of the reference set of characteristics, as the characteristics are designed and defined in enough detail to implement them. Additionally, these features are used to define a similarity distance between multi-objective optimization problems.

The rest of this chapter is structured as follows: Section 4.2 provides a literature review on the state of the field of landscape characterization for optimization problems. In Section 4.3, the selected landscape characteristics, and the implementations details on their computation, are discussed. An evaluation on the stability and characterization capabilities of the selected set of characteristics is provided in Section 4.4.

[1] From The American Heritage®Dictionary of the English Language, 5th Edition.
[2] Available at: `https://gitlab.com/jfaldanam-phd/moorphology`

Type	Name	Description
Problem-dependent		
	name	Name of the problem being sampled
	n_var	Number of variables
	n_obj	Number of objectives
	n_cons	Number of constraints
	sample_size	Number of samples used to extract the landscape characteristics
Global		
	dist_x_avg	Average distance among solutions in the variable space
	dist_x_max	Maximum distance among solutions in the variable space
	dist_f_avg	Average distance among solutions in the objective space
	dist_f_max	Maximum distance among solutions in the objective space
	nd_n	Proportion of non-dominated solutions
	dist_x_nd_avg	Average distance among non-dominated solutions in the variable space
	dist_x_nd_max	Maximum distance among non-dominated solutions in the variable space
	rank_avg	Average rank w.r.t. non-dominated sorting
	rank_max	Maximum rank w.r.t. non-dominated sorting
	rank_ent	Entropy of the number of solutions per rank w.r.t. non-dominated sorting
Evolvability		
	sup_avg_neig	Average proportion of dominating neighbours
	inf_avg_neig	Average proportion of dominated neighbours
	inc_avg_neig	Average proportion of incomparable neighbours
	lnd_avg_neig	Average proportion of locally non-dominated neighbours
	lsupp_avg_neig	Average proportion of supported locally non-dominated neighbours
	dist_x_avg_neig	Average distance from neighbours in the variable space
	dist_x_max_neig	Maximum distance from neighbours in the variable space
	dist_f_avg_neig	Average distance from neighbours in the objective space
	dist_f_max_neig	Maximum distance from neighbours in the objective space
Ruggedness		
	dist_x_cor_neig	Neighbour's correlation of the average distance from neigh. in the variable space
	dist_f_cor_neig	Neighbour's correlation of the average distance from neigh. in the objective space

Table 4.1: Selected landscape characteristics with their definition. Most of them are originally defined in [94].

4.2 Related works

As mentioned in the previous section, [94] is the main inspiration for this work as it defines the set of characteristics that have been implemented inside *moorphology*. These characteristics are also known as "meta-features", which are described as a set of data points that characterize problem properties and, more importantly, their relations with algorithm performance [27, 93].

Cosson et al. [30] provides a deep study on the impact of sampling on the extraction of multi-objective landscape features, and their integration into feature-informed performance prediction and algorithm selection approaches, with a focus in combinatorial multi-objective landscapes. This work shows that a random walk using a reasonably small proportion of neighbors leads to cheap and informative feature values. This type of sampling is taken into account in the characteristics used in this work.

However, none of them provide an open source practical implementation that is readily available for use. *moorphology* fills that gap for the jMetal framework.

4.3 Implementation of the characteristics to study

The landscape features proposed by [30] are based on measurements calculated over a set of solutions obtained through a sampling of the problem under study. The set of characteristics has been filtered to include only those whose correlation with algorithm performance, in absolute terms, is greater than 0.5, according to the reference article. Table 4.1 displays the selected characteristics computed by *moorphology*.

The sampling process utilizes random Latin hypercube sampling [106] to select $n = 200 \cdot d$ distinct solutions, where d represents the number of variables in the problem. For each solution (x) within the sampling set (P), denoted as $x \in P$, its d closest solutions, measured by the Euclidean distance in the variable space from the sampling set excluding x, are considered its neighbors. The Spearman correlation [144] is employed by all metrics that involve correlations. To enable comparison across different problems, all characteristics related to distance (except dist_x_max and dist_f_max) are normalized using the max-min method based on the maximum and minimum values within their respective search spaces, whether variable or objective space, for that particular sample [65]. The utilization of Latin hypercube sampling ensures that some of the samples are proximate to the true maximum and minimum values.

moorphology is a Maven project develop using Java 17 and jMetal 6.1, using best DevOps practices for for automated testing and releases via continuous integration and continuous delivery (CI/CD) pipelines. Following the implementation details mentioned above, *moorphology* implements them to be compatible to any multi-objective problem that follows the jMetal interface for continuous problems, including those with an unknown Pareto front. The return characteristics are stored on JSON format and an example of the output can be seen in Listing 4.1.

4.4 Evaluation of the selected set of characteristics

To evaluate the characteristic selected and their implementation, two evaluations methods are proposed. Firstly, an analysis on the stability of the selected characteristics is implemented, using the COCO bi-objective functions [20] as the suite of problems to study. COCO (Comparing Continuous Optimizers) is a platform

```
{
  "RE21": {
    "sampleSize": 800,
    "numberOfObjectives": 2,
    "numberOfVariables": 4,
    "distanceXMaximumAnalytically": 3.6096311794313363,
    "distanceXMaximum": 3.191955229900468,
    "distanceXAverage": 0.4380010221881074 4,
    "distanceFMaximum": 1483.6531068237502,
    "distanceFAverage": 0.2211367795028989,
    "distanceXNonDominatedMaximum": 3.0074930957408235,
    "distanceXNonDominatedAverage": 194.17125218574452,
    "neighbourDistanceXMaximum": 0.4456562249147494,
    "neighbourDistanceXAverage": 1.238361757886079E-4,
    "neighbourDistanceFMaximum": 146.25549302153237,
    "neighbourDistanceFAverage": 5.8994922707436544E-5,
    "averageProportionOfDominatingNeighbours": 0.235625,
    "averageProportionOfDominatedNeighbours": 0.205625,
    "averageProportionOfIncomparableNeighbours": 0.30875,
    "averageProportionOfLocallyNonDominatedNeighbours": 0.4209375,
    "averageProportionOfSuportedLocallyNonDominatedNeighbours": 0.1584375,
    "neighboursCorrelationOfAverageDistanceX": 0.081875,
    "neighboursCorrelationOfAverageDistanceF": 0.159375,
    "proportionOfNonDominated": 0.03875,
    "rankMaximum": 2,
    "rankAverage": 0.3225,
    "rankEntropy": 1.0190924654082565
  },
  "RE91": {
    "sampleSize": 1400,
    "numberOfObjectives": 9,
    "numberOfVariables": 7,
    "distanceXMaximumAnalytically": 2.6499245272271437,
    "distanceXMaximum": 2.2544270245973133,
    "distanceXAverage": 0.46482071474670555,
    "distanceFMaximum": 161.90958649715395,
    "distanceFAverage": 0.16001637801620344,
    "distanceXNonDominatedMaximum": 2.2544270245973133,
    "distanceXNonDominatedAverage": 330.4709386002521,
    "neighbourDistanceXMaximum": 0.534838024563182,
    "neighbourDistanceXAverage": 1.068143117153318E-4,
    "neighbourDistanceFMaximum": 132.32731806260333,
    "neighbourDistanceFAverage": 8.52498461288246E-5,
    "averageProportionOfDominatingNeighbours": 0.054591836734693866,
    "averageProportionOfDominatedNeighbours": 0.03887755102040838,
    "averageProportionOfIncomparableNeighbours": 0.763673469387763,
    "averageProportionOfLocallyNonDominatedNeighbours": 0.8222448979591944,
    "averageProportionOfSuportedLocallyNonDominatedNeighbours": 0.013979591836734678,
    "neighboursCorrelationOfAverageDistanceX": 0.04530612244897945,
    "neighboursCorrelationOfAverageDistanceF": 0.13918367346938756,
    "proportionOfNonDominated": 0.6264285714285714,
    "rankMaximum": 1,
    "rankAverage": 0.018571428571428572,
    "rankEntropy": 0.13334260218394456
  }
}
```


Listing 4.1: Output of *moorphology* for the characterization of the landscape of the RE21 and RE91 multi-objective problems.

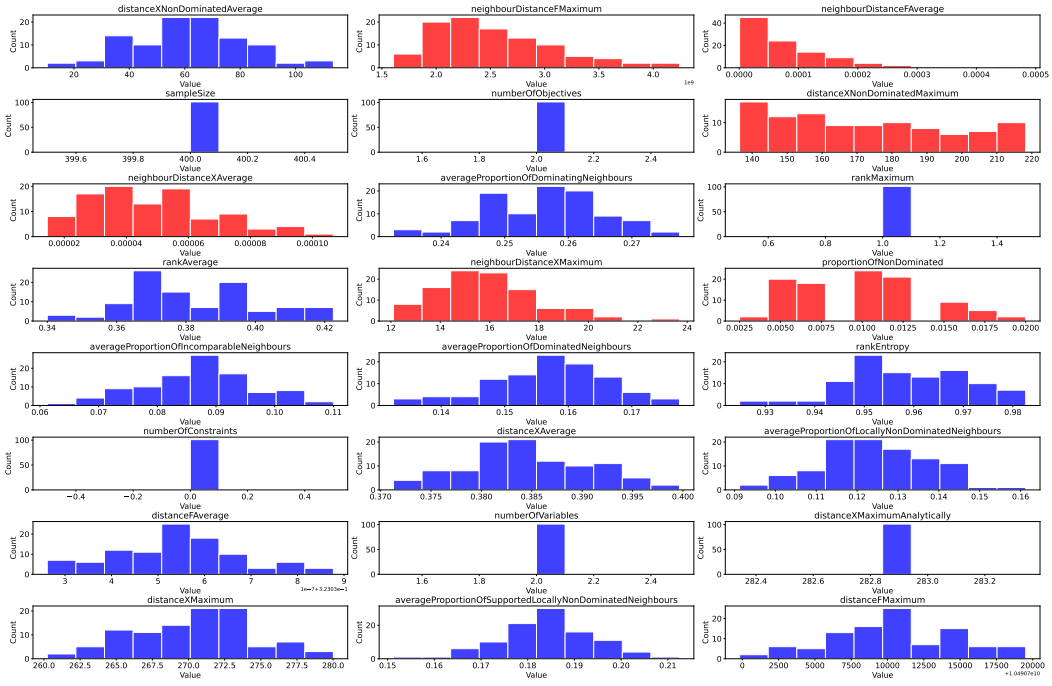


Figure 4.1: Shapiro-Wilk normality test applied to the COCO instance formed by functions *f001* and *f002* of the bbob-biobj problem suite. Graphs in blue means the characteristic follows a normal distribution according to the Shapiro-Wilk normality test, red means it does not.

to benchmark and compare continuous optimizers, also known as non-linear solvers for numerical optimization [61]. The bi-objective *bbob-biobj* test suite is COCO's first multi-objective test suite with 55 noiseless, scalable bi-objective functions that come from the combination of 10 mono-objetive functions of the original *bbob* that include separable, unimodal and multimodal functions.

Secondly, a similarity measurement between different multi-objective problems is defined and later evaluated against expert knowledge to verify that the set of characteristics is expressive enough for the objectives of this PhD thesis.

4.4.1 Evaluation of the stability over the COCO bi-objective suite

In this section, a statistical analysis on the selected characteristics is applied to verify the stability of their results. If some of the characteristics are not stable (their values are too dependent on the specific sampling obtained), they are not

Characteristic	Mean	Stdev	Variance	Is gaussian	P-value	Stat
inf_avg_neig	0.15698	0.00924	8.5352e-05	Yes	0.46918	0.98756
sup_avg_neig	0.25605	0.00892	7.9492e-05	Yes	0.79468	0.99171
inc_avg_neig	0.08697	0.00954	9.1105e-05	Yes	0.15496	0.98102
lnd_avg_neig	0.12441	0.01298	0.00016849	Yes	0.72547	0.99083
lsupp_avg_neig	0.18427	0.01037	0.00010763	Yes	0.85161	0.99249
dist_f_avg	0.32303	1.3556e-07	1.8375e-14	Yes	0.26639	0.98409
dist_f_max	1.0491e+10	4372.89	1.9122e+07	Yes	0.38298	0.98626
dist_x_avg	0.38465	0.00608	3.7005e-05	Yes	0.64535	0.98985
dist_x_max	270.28	4.1822	17.491	Yes	0.67746	0.99024
dist_x_nd_avg	61.211	19.368	375.13	Yes	0.84707	0.99243
dist_x_nd_max	171.61	24.835	616.80	No	8.8167e-05	0.93491
dist_f_avg_neig	8.4385e-05	8.7613e-05	7.6760e-09	No	5.6009e-10	0.81317
dist_f_max_neig	2.5436e+09	5.4518e+08	2.9722e+17	No	0.00054	0.94760
dist_x_avg_neig	4.7941e-05	2.0107e-05	4.0427e-10	No	0.00978	0.96561
dist_x_max_neig	15.865	2.0422	4.1705	No	0.00173	0.95516
n_cons	0	0.0	0.0	Yes	1.0	1.0
n_obj	2	0.0	0.0	Yes	1.0	1.0
n_var	2	0.0	0.0	Yes	1.0	1.0
nd_n	0.00995	0.00395	1.5623e-05	No	0.00030	0.94362
rank_avg	0.38233	0.01836	0.00033716	Yes	0.07423	0.97697
rank_ent	0.95865	0.01253	0.00015700	Yes	0.15355	0.98097
rank_max	1	0.0	0.0	Yes	1.0	1.0
sample_size	400	0.0	0.0	Yes	1.0	1.0

Table 4.2: Summary of Statistical Analysis for all characteristics for the problem formed by functions *f001* and *f002* of the bbob-biobj problem suite.

interesting for the purposes of this thesis.

To evaluate the stability of the characteristics, first, for each of the 55 COCO multi-objective problems, 100 independent samplings are computed. Secondly, the 100 samplings for each problem are evaluated by the Shapiro-Wilk [136] normality test, and, finally, the mean, variance and standard deviation are calculated for each pair, characteristic-problem, that follows a normal distribution. These metrics provide the quantitative data required for evaluating the stability of the proposed characteristics.

Figure 4.1 showcases the visual representation of the normality test results for a specific problem (bbob_f001_i03_d02__bbob_f002_i05_d02), while Table 4.2

shows the statistical results.

From the experiment the following conclusions could be extracted. Firstly, landscape characteristics related to the distance with other solutions of the neighborhood do not follow a normal distribution and, as such, are not ideal for comparing the similarity of two problems. This is due to the inherent randomness between several samplings. For example, on the maximum distance on the variable space between the non-dominated solutions, for a specific problem in Figure 4.1, the values are very spread across the range of possible values. Secondly, some of the characteristics are deterministic, such as the number of objectives, variables and constraints, the maximum distance in variable space when calculated analytically and the maximum rank. The determinism of this characteristics can prove useful for filtering problems before doing a similarity search, as defined in the following section.

4.4.2 Evaluation on the similarity between problems

This section provides a qualitative evaluation of the set of characteristics implemented by defining a similarity metric between multi-objective problems based on said characteristics and comparing it to previous experiences from experts of the field.

The similitude between problems is defined as the sum of absolute differences between normalized values of corresponding characteristics of the two problems. The normalization is done by dividing the smaller value by the larger one for each characteristic, subtracting this ratio from 1, and then taking the absolute value. This distance can be formally defined as follows:

Given two problems, P_1 and P_2, and a set of characteristics $C = \{c_1, c_2, ..., c_n\}$ where c_i is a characteristic (as shown in Table 4.1), the distance $D(P_1, P_2)$ between the problems is computed using Equation 4.1.

$$D(P_1, P_2) = \sum_{i=1}^{n} \left| 1 - \frac{\min(c_i(P_1), c_i(P_2))}{\max(c_i(P_1), c_i(P_2))} \right| \tag{4.1}$$

Here, $c_i(P)$ denotes the value of characteristic c_i for problem P. The function $\min$ and $\max$ provide the smaller and larger values, respectively, between $c_i(P_1)$ and $c_i(P_2)$. The closer this metric is to 0, the more similar the two problems are in terms of their topology. An implementation of this distance in SPARQL is provided in Listing B.6.

Figure 4.2 shows the similarity of each problem in the DTLZ, GLT, LZ09, UF,

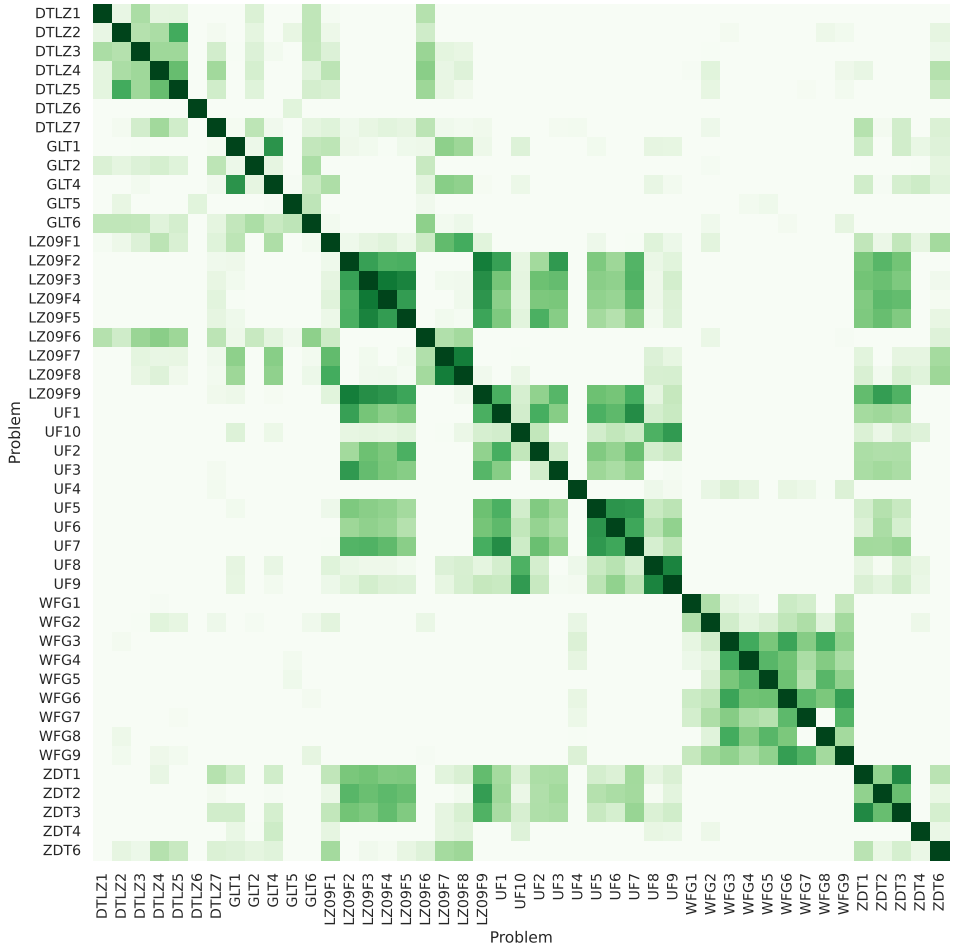


Figure 4.2: Similitude for the problems of the DTLZ, GLT, LZ09, UF, WFG and ZDT family by using the defined metrics.

WFG and ZDT families against the rest of the studied problems. The results looks promising, as the diagonal shows a strong similarity, that means that every problem is similar to itself, and there are clear clusters of similar problems inside most families of problems. These clusters suggest that authors of benchmark problems introduce biases inside the different problems of the same family, maybe due to reusing of functions between different problems.

Other important details are the similarities between some of the different families of problems. Firstly, we can see that LZ09 and UF problem family have some similarity between them. A further study indicates that some of the LZ09 where later reuse in the UF problem family, for the CEC 2009 competition [170]. Also, the LZ09 problem families and the ZDT 1-3 problems are also similar. This fact may be due to how they share the same Pareto front, even though the landscape to get there is unrelated.

To delve deeper into a smaller set of problems, let's focus on the ZDT problem family. It is known that the ZDT4 behaves differently than all others, as it is the only multimodal problem in its family [112] and ZDT6 as the solutions of the Pareto front are not uniformly separated. This is reflected in the similitude matrix, as ZDT4 and ZDT6 are the least similar to the rest. To enter in more detail on the similitude inside this family of problems, Figure 4.3 shows the similarity between the ZDT family problems in each of the landscape characteristics defined using as reference the ZDT1 and ZDT4 problems.

When using the ZDT1 as reference, the main difference observed with the ZDT4 and ZDT6 problems are in the absolute distance related metrics, both in search and objective spaces. When using as reference the ZDT4, a similar relation can be seen, but in this case the ZDT4 problem is not close to any of the others. From these plots, the problem family can be split in 3 groups according to their landscape, ZDT 1-3 as they are fairly similar between them, and then both ZDT4 and ZDT6 in a different group each, as they are do not share the distance related metrics with any of the others.

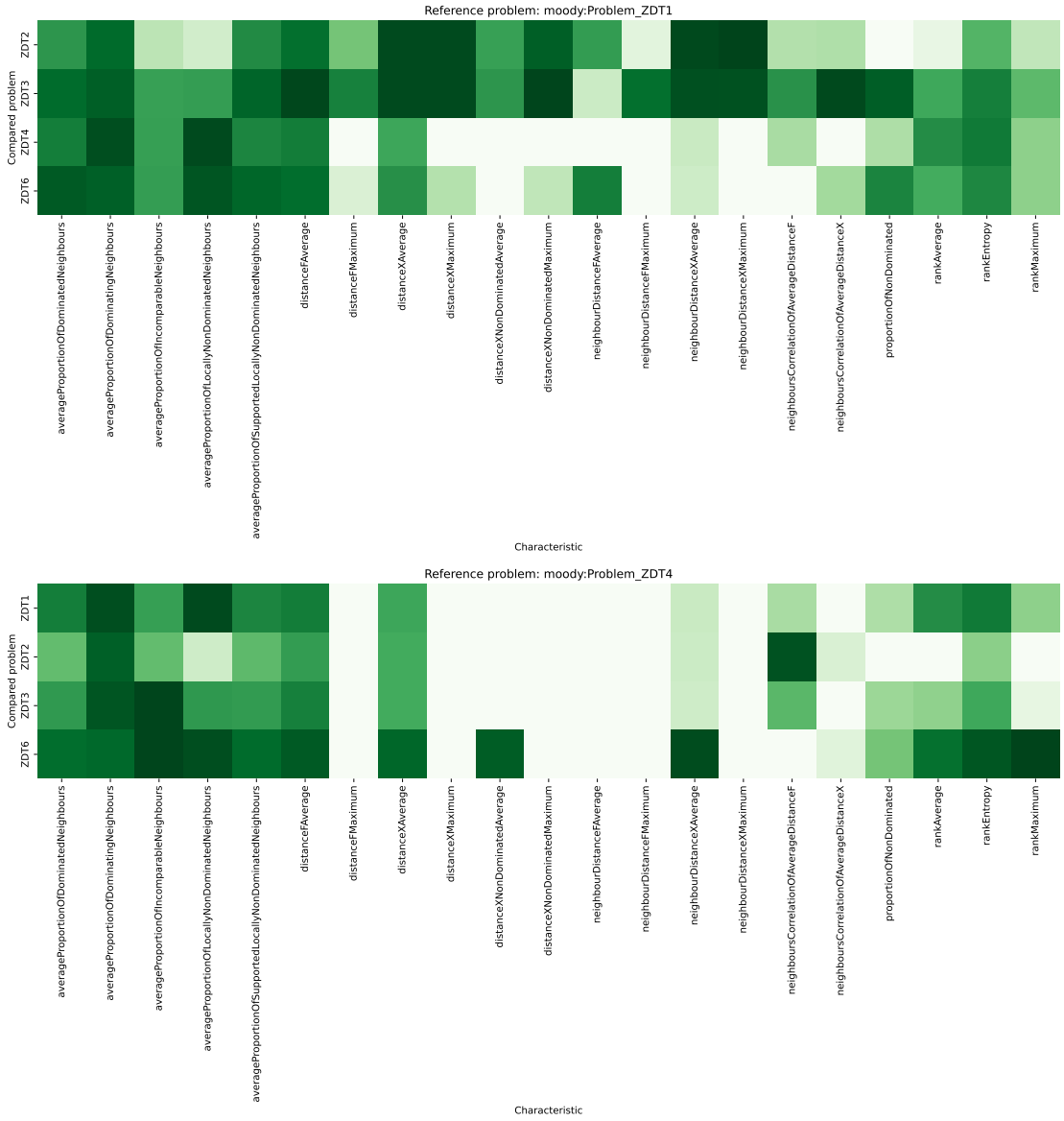


Figure 4.3: Similarity between the problems of the ZDT family in each characteristi, using ZDT1 (top) or ZDT4 (bottom) as references.

Chapter 5

Automatic Generation of Quality Algorithmic Configurations for Metaheuristics

To automatically generate suitable configurations for multi-objective metaheuristics, this chapter proposes to approach the auto-configuration of multi-objective optimization algorithms by meta-optimization, that is using a metaheuristic to optimize the parameters of another metaheuristic. The idea is to formulate the configuration of a multi-objective metaheuristic as an optimization problem where the decision variables are the algorithm parameters and the objectives are defined as quality indicators.

A study on using the NSGA-II algorithm as an automatic configuration tool to find configurations for the NSGA-II to validate the proposed approach. Then, a new auto-configuration tool, *Evolver*, utilizing this approach is presented based on the jMetal framework. *Evolver* is used to populate the recommendation system proposed in this PhD thesis.

5.1 Introduction

As previously mentioned, the quality of the Pareto front approximations found by multi-objective evolutionary algorithms is affected by the values of their control parameters. This means that, given a set of problems to be optimized and a given algorithm, the user has to fine tune the algorithms parameters to get accurate results. The approach commonly adopted to carry out this task is to try to adjust the parameters manually by conducting pilot tests, which is a trial-and-error strategy. Furthermore, this process requires knowledge of the algorithm, which users who are experts in other domains do not usually possess. The consequence is that those users are likely to end up selecting a well-known algorithm, typically NSGA-II [35], with default settings.

In this context, the recommendation system described in this PhD thesis aims to help those users to find suitable configurations to solve their problems. However, the recommendation system requires enough evaluated configurations to power the recommendations. To solve the problems of finding automatically suitable configurations to solve specific problems, an active research line is automatic algorithm configuration [73], consisting in taking a set of problems as training set to find a particular parameter configuration of the parameters to a produce version of the algorithm that, configured with them, can solve those problems efficiently. An extension of this idea is automatic algorithm design, where not only parameters but also algorithmic components can be combined to design a new algorithmic variant. An advantage of these approaches is that they can be supported by tools that help to find the configurations automatically, such as irace [103], paramILS [77], GSF [166] and SMAC3 [96]. Focusing on multi-objective evolutionary algorithms, irace has been applied in several works [11, 12, 115].

In this chapter, a study is conducted about the use of NSGA-II to find configurations of NSGA-II, i.e., using NSGA-II as meta-optimizer [117]. The basic idea is to consider the auto-design of NSGA-II as a multi-objective problem, where the decision variables represent parameters and components and the objectives can be combinations of quality indicators [174].

This motivation stems, first, from previous experiences in automatic design of multi-objective metaheuristics, which are based on combining the jMetal optimization framework [41, 113] with irace to find configurations of NSGA-II [43, 114] and particle swarm optimizers [40]. Second, a recent survey [73] that remarked as future research prospects easy-to-use algorithm tuning and multi-objective approaches.

This study intends to answer these two research questions:

- RQ1: how complex is to build the meta-optimization package?. The objective is a simple and easy-to-use software solution.
- RQ2: can accurate configurations be found? The search capabilities of the meta-optimizer must be validated by conducting representative experiments.

Following the validation of this approach, *Evolver* [1] is introduced as a tool aimed at automatically configuring and designing metaheuristics for solving multi-objective optimization problems [28]. This approach leverages the similarities between the optimization problems faced by metaheuristics and the problem of configuring and designing them.

Evolver is implemented in Java as a Maven package and built upon jMetal [41, 113], a framework for multi-objective optimization using metaheuristics that implements highly configurable versions of representative multi-objective solvers. These solvers include representative algorithms of three types of multi-objective evolutionary algorithms: NSGA-II [35], MOEA/D [89], and SMS-EMOA. These algorithms represent approaches based on Pareto dominance, decomposition, and quality indicators, respectively.

Integrating *Evolver* with jMetal gives users access to a wide range of algorithms that can be used as meta-optimizers as well as to a large amount of benchmark problems (the current version of *Evolver* is aimed at solving continuous problems); any continuous problem implemented with jMetal can be used by *Evolver*. Additionally, jMetal offers a diverse set of quality indicators that can measure the solution quality of a multi-objective problem according to different properties, such as convergence and diversity in the decision space. These indicators can be used in various combinations as optimization objectives of a meta-optimizer. As a result, *Evolver* allows researchers to study the effectiveness of different meta-optimizers and the influence of using varied objectives when configuring and designing metaheuristics. Additionally, the proposed tool has a user-friendly graphical interface that simplifies the selection and execution of both the meta-optimizer and the algorithm to be configured, along with the choice of optimization goals.

The rest of this chapter is organized as follows. First, Section 5.2 provides a review of related works in the literature. Section 5.3 provides a detailed description of the presented approach for the automated design of a meta-optimizer for NSGA-II. In Section 5.4, the results of the study conducted to validate this

[1] Available at: `https://github.com/jMetal/Evolver`

proposal are presented. The findings and implications of these experiments are discussed in Section 5.5.

After the validation of the proposal, Section 5.6 delves into the development of a software tool for meta-optimization, *Evolver*. Section 5.7 shows an example of how a end-user could use *Evolver*, while Section 5.8 discusses how this tool can impact researchers. Finally, Section 5.9 proposes a different promising approach and sets the stage for future research beyond the scope of this PhD thesis.

5.2 Related works

A recent survey on automatic parameter tuning methods for metaheuristics [73] analyzed the advantages and shortcomings of different methods, and highlighted research prospects in the field. The survey noted the need for an easy-to-use algorithm tuning toolbox, as most existing auto-configuration tools require the use of the command line and scripts, lacking a visual user interface. Additionally, the survey indicated that a challenge that is worth to be studied is multi-objective tuning approaches, which involve optimizing more than one performance metric simultaneously.

When focusing on multi-objective optimization, automatic parameter tuning has been applied to multi-objective variants of ant colony optimization [102], evolutionary algorithms [11, 121, 13], and particle swarm optimization [95, 40]. While some of these studies are based on ad hoc techniques, the upsurge of automatic parameter configuration and design tools is proving useful for practitioners. Notable packages include irace [103], SMAC3 [96], and ParamILS [77]. These tools are designed to deal with a single performance objective, while SMAC3 (since version 1.2) and the multi-objective extension of ParamILS (MO-ParamILS[16]) support the optimization of two or more competing performance indicators.

Comparing *Evolver* to MO-ParamILS and SMAC3, there are noticeable differences. MO-ParamILS utilizes a multi-objective iterated local search process with a non-dominated configurations archive, while *Evolver* supports a wide range of multi-objective metaheuristics. SMAC3 uses a Bayesian Optimization and utilizes a mean aggregation strategy, aggregating several objectives into a single scalar objective that is then optimized by SMAC3, while the metaheuristics of *Evolver* yield as a result fronts of non-dominated solutions according to the defined objectives. Furthermore, *Evolver* is designed for the specific task of finding configurations of multi-objective metaheuristics, making use of the large set of resources (algorithms, problems, quality indicators) that are not provided by

Parameter/Component	Type	Domain	Dependency
algorithmResult	c	{EA, population}	
populationSizeWithArchive	i	[10, 200]	algorithmResult == EA
externalArchive	c	{CD, unbounded}	algorithmResult == EA
offspringPopulationSize	i	[1, 400]	
selection	c	{tournament, random}	
selectionTournamentSize	i	[2, 10]	selection == tournament
createInitialSolutions	c	{random, LHS, scatterSearch}	
crossover	c	{SBX, BLX_ALPHA, WA}	
crossoverProbability	r	[0.0, 1.0]	
crossoverRepairStrategy	c	{random, round, bounds}	
sbxDistributionIndex	r	[5.0, 400.0]	crossover == SBX
blxAlphaCrossoverAlphaValue	r	[0.0, 1.0]	crossover == BLX_ALPHA
mutation	c	{uniform, polynomial, LP, NU}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	mutation $\in$ {polynomial, LP}
uniformMutationPerturbation	r	[0.0, 1.0]	mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	mutation == NU

Table 5.1: Design space of the configurable NSGA-II algorithms in *Evolver*. Types: (c)ategorical, (i)nteger, (r)eal. (EA; externalArchive, CD; crowdingDistanceArchive, LHS; latinHypercubeSampling, WA; wholeArithmetic, LP; linkedPolynomial, NU; nonUniform)

other frameworks.

5.3 Meta-optimization approach

The process of auto-designing evolutionary algorithms requires three elements: the design space, an algorithmic template, and an auto-design tool. These elements are described next, including the approach to handling them.

5.3.1 Design space

The design space is composed of the algorithm parameters and components, their types, allowed values, and, optionally, constraints. In the case of NSGA-II, consider a flexible definition of it, in which a multi-objective evolutionary algorithm adopting a replacement strategy based on dominance ranking and the crowding distance density estimator is considered a NSGA-II variant. Table 5.1 defines the design space of this NSGA-II, which is similar to the ones used in former works [115, 114] (please refer to these references for a detailed explanation of the parameters and components).

An example of parameter is the offspring population size, which is an inte-

ger variable taking values in the range [1, 400] (a value of 1 would lead to a steady-state version of NSGA-II). Examples of components are the crossover and mutation operators. Each operators can in turn also have specific parameters, such as the distribution index for SBX crossover.

A design decision is whether NSGA-II uses an external archive (i.e., an auxiliary population) or not. If the population size is P, the idea is that any evaluated solution is inserted into the archive, which keeps only non-dominated solutions, and the result of the algorithm would be P solutions from the archive; in this case, the population size is not fixed and it can take a value between 10 and 200. The external archive can be bounded (the crowding distance is used as density estimator to remove solutions when the archive size is greater than P) or unbounded (in this case, all the evaluated solutions are inserted and, when the algorithm finishes, P evenly spread solutions are returned).

5.3.2 Algorithmic template

Since release 6.0, jMetal includes a *jmetal-auto* package containing an implementation of NSGA-II, called AutoNSGAII, which can take any valid combination of the parameters and components of Table 5.1, generating different NSGA-II versions.

The input of AutoNSGAII is a string containing all the parameter names and their values. This string is parsed internally and AutoNSGAII is configured with the parameter values and the components specified in the string. An example of a subset of this string is the following: "*–archiveResult externalArchive –offpringPopulation 40 –selection tournament ...*"

5.3.3 Meta-optimizer

In previous works from the literature combining jMetal with irace [43, 114], the finding of configurations is based on running irace, which generates combinations of valid configurations according to the design space. For each configuration, irace runs AutoNSGAII, which returns as a result the value of a quality indicator; this value is taken by irace as a measure of the quality of the configuration.

As the goal is to replace irace by the NSGA-II algorithm implemented in jMetal, which would act as meta-optimizer, it is necessary to formulate and implement the optimization problem that would to be solved by the meta-optimizer. This problem has the following parameters:

- List of problems used as training set.
- List of quality indicators, being each indicator an objective to be minimized.
- The population size of AutoNSGAII.
- The stopping condition of AutoNSGAII (in terms of number of evaluations).
- Number of independent runs of AutoNSGAII for each configuration to be evaluated.

To define the problem encoding, the approach adopted is simple: every parameter of Table 5.1 is represented as a real value in the range $[0.0, 1.0]$, so the solutions are composed of 18 decision variables. When a solution has to be evaluated, the variables are decoded to construct the parameter string that is used when calling AutoNSGAII. The decoding is done as follows:

- Real parameter: the value is scaled up from $[0.0, 1.0]$ to the range of the parameter (e.g.,. $[5.0, 400.0]$ in the case of the SBX distribution index).
- Integer parameter: same procedure as for real parameters, but the resulting value is truncated.
- Categorical parameter: the interval $[0, 0, 1, 0]$ is divided into sub-intervals according to the number of parameter values, and the index of the sub-interval is used to obtain the actual categorical value.

Once the parameter string is decoded, AutoNSGAII is called to solve all the problems of the training set as many times as the number of independent runs. For each obtained front, the quality indicators are computed and the resulting objectives values of evaluating a configuration is the median of the median of the quality indicators of all the problems of the training set.

Let's focus on the pros and cons of this approach. Starting by the cons, parameter constraints are not being considered, so all the elements of design space are included although some of them may be ignored (e.g., the uniform perturbation is useless if the selected mutation operator is polynomial), and the discretization of categorical parameters using sub-intervals can lead to different solutions being equivalent if all variables have the same values except one corresponding to a categorical parameter whose values are in the same sub-interval. As advantages, the encoding is very simple and any multi-objective algorithm in jMetal able of solving continuous problems can be used as meta-optimizer.

5.4 Experimental study

The intention is to empirically validate this approach through a series of experiments, which are categorized into two distinct scenarios. These experiments are described below, detailing their purpose, expected outcomes and results.

The meta-optimizer is configured with the additive epsilon (EP) and normalized hypervolume (NHV) quality indicators as the objective functions to be minimized. The first indicator measures the convergence of a Pareto front approximation while the second one takes into account both convergence and diversity [174]. NHV is used instead of plain hypervolume as for this latter, the bigger its value the better, while jMetal minimizes objective functions by default. NHV is defined as 1.0 minus the hypervolume of the front divided by the hypervolume of the reference front.

The meta-optimizer has been configured using common parameter values associated with NSGA-II. The population size is 50 and the variation operators are SBX crossover (with probability 0.9 and a distribution index value of 20.0) and polynomial mutation (with probability $1/n$, being n the number of decision variables of the problem, and a distribution index value of 20.0). The stopping condition is set to 3000 function evaluations. The NSGA-II implementation in jMetal can be executed in parallel both using a synchronous or an asynchronous scheme [42].

Next, let's define two scenarios and three experiments.

5.4.1 Scenario 1: Finding Configurations for Single Problems

The first scenario is aimed at determining whether an meta-optimization approach is able of finding well-performing configurations of NSGA-II for single problems. For that, experiments focus on two problems:

- **Experiment 1 - problem ZDT4**: this problem [175] is a bi-objective multi-frontal problem, whose default configuration consists of 10 decision variables. The standard NSGA-II has difficulty in providing Pareto front approximations with a uniform spread of solutions. Previous studies [43] have shown that using a steady-approach can significantly improve the diversity of the fronts. Pilot tests also indicate that comparable improvements can be achieved when using an external bounded archive.
- **Experiment 2 - problem DTLZ3**: This problem belongs to the DTLZ benchmark [36]. It is formulated with a default configuration consisting of twelve decision variables and three objectives. DTLZ3 is also a multi-modal

problem with a convex Pareto front. The study presented in [115] showed that both, NSGA-II and the AutoNSGAII, configured with irace were unable to find accurate approximated fronts in terms of convergence and diversity for this problem. According to other works [79], NSGA-II is able of finding accurate fronts for problem DTLZ2 when using an external unbounded archive and retrieving from it a subset of evenly distributed solutions. DLTZ2 is not multi-modal but shares many similarities with DTLZ3 (i.e., convex Pareto front, three objectives, and twelve decision variables). The objective here is twofold: 1) to determine whether the meta-optimizer is able of finding a configuration to effectively solve DTLZ3; and, 2) to check if that configuration includes an unbounded archive.

5.4.2 Scenario 2: Finding Configurations for Sets of Problems

The above scenario must validate the potential of auto-configuration applied to optimize single problems. The found configurations may be, however, too specific to that particular problem, and perform poorly for other problems (overfitting). The second scenario addresses this issue by auto-configuring the algorithm on a set of problems instead of just one. Additionally, the obtained configurations are validated using different sets of problems.

- **Experiment 3 - WFG benchmark**: This experiment aims to replicate the study presented in [115]. NSGA-II is configured for optimizing the nine problems of the WFG suite [75], which are *training set*. The found configurations are later used to solve both the WFG problems and the seven instances of the DTLZ family problems–the *validation set*. All the problems in this experiment are formulated as bi-objective ones.

5.4.3 Results

This section reports and analyze the results obtained on the three defined experiments. In all the cases, the number of independent runs per configuration is set to 3.

Experiment 1

The stopping condition of AutoNSGAII is set to 15000 function evaluations. Figure 5.1 shows computed fronts by the meta-optimizer at 1000, 2000, and 3000 evaluations. As shown, the final front is composed of only one solution, and the figure suggests that the meta-optimizer might not have converged in the performed evaluations. The found design in this experiment (see Table 5.2)

Parameter	NSGA-II	Exp. 1	Exp. 2	Exp. 3
populationSize	100	100	100	100
createInitialSolutions	random	LHS	scatterSearch	random
algorithmResult	population	externalArchive	externalArchive	externalArchive
externalArchive	-	CD	unboundedArchive	CD
populationSizeWithArchive	-	106	58	61
offspringPopulationSize	100	60	130	68
crossover	SBX	SBX	SBX	BLX_ALPHA
crossoverProbability	0.9	0.991	0.942	0.858
crossoverRepairStrategy	random	round	random	bounds
sbxDistributionIndexValue	20.0	5.11	70.479	-
blxAlphaCrossoverAlphaValue	-	-	-	0.547
mutation	polynomial	polynomial	uniform	linkedPolynomial
mutationProbabilityFactor	1	0.76	0.699	0.161
mutationRepairStrategy	random	bounds	round	round
PMDI	20	32.23	-	11.335
uniformMutationPerturbation	-	-	0.417	-
selection	tournament	tournament	random	tournament
selectionTournamentSize	2	9	-	4

Table 5.2: Best configuration found for the NSGA-II on each experiment. (LHS; latinHypercubeSampling, CD; crowdingDistanceArchive, PMDI; polynomialMutationDistributionIndex)

includes a bounded external archive with crowding distance; as commented before, the use of this kind of archive is known to be beneficial for converging and for achieving a front of evenly spread solutions.

AutoNSGAII with the obtained configuration is compared with NSGA-II with default settings next. The stopping condition is set to 25000 function evaluations in both cases and compare the Pareto front approximations computed by both algorithms. The front computed with the found configuration (Figure 5.2 right) has a noticeable better convergence and spread than the approximation computed by NSGA-II with standard the default setting (Figure 5.2 left).

Experiment 2

In this experiment, the stopping criterion for AutoNSGAII has been raised to 20000 evaluations. Figure 5.3 shows the approximation fronts computed after 1000, 2000 and 3000 evaluations. In this case, the figure suggest that the meta-optimizer has almost converged after performing the 3000 function evaluations. The computed approximation front consists of twelve points. The configuration corresponding to the point with the lowest NHV value (on the right end) is included in Table 5.2. As expected, the configuration found by the meta-optimizer uses the unbounded external archive.

Figure 5.4 compares the approximation front computed with NSGA-II and

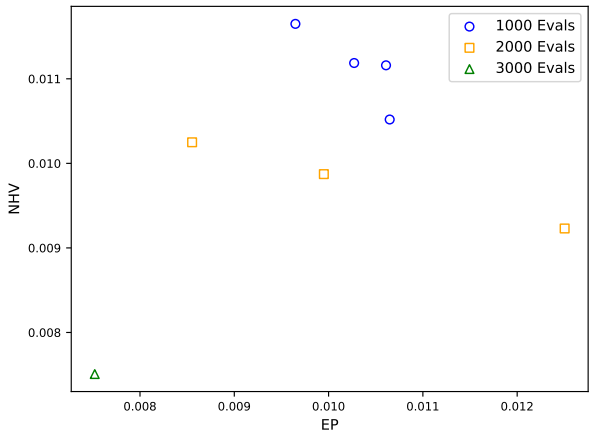


Figure 5.1: Problem ZDT4. Evolution of the front generated by the meta-optimizer.

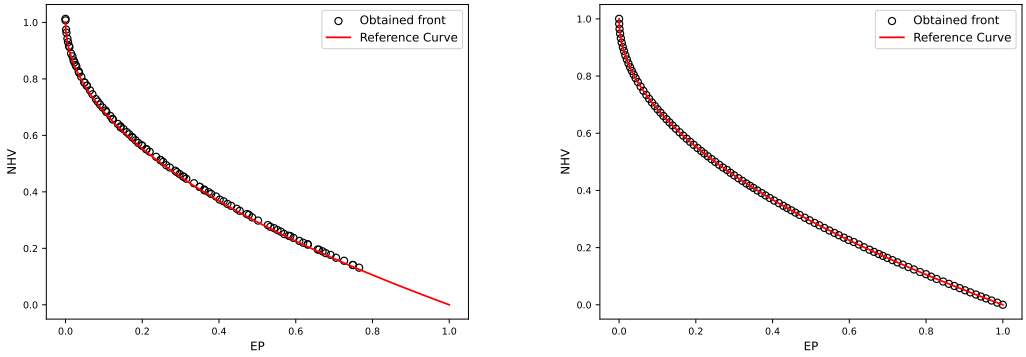


Figure 5.2: Problem ZDT4. Pareto front approximation found by the standard NSGA-II (left), and Pareto front approximation found by the auto-designed NSGA-II (right).

the one computed with the configuration found the meta-optimizer using AutoNSGAII. In both cases, 40000 function evaluations is used as the stopping criterion. The graph shows remarkable differences between the front computed by NSGA-II (poor convergence and coverage of the Pareto front approximation) and AutoNSGAII.

Experiment 3

In alignment with existing work [115], the stopping criterion of AutoNSGAII is set to 25000 evaluations for this experiment. The evolution of the fronts over different number of evaluations is shown in Figure 5.5. As in the previous experiment, the point with the minimum NHV value is taken and its corresponding configuration is used to compare with the results reported in [115]. The comparison in this case includes NSGA-II, and SMPSO [110] with their default settings and AutoNSGAII with the mentioned configuration.

The chosen configuration from the meta-optimizer is summarized in Table 5.2. Interestingly, this configuration is similar to the one computed in [115]: both share the use use BLX_ALPHA crossover and an external bounded archive).

Table 5.3 showcases the validation results of the auto-designed NSGA-II for the WFG and DTLZ benchmarks. Tables (a) and (b) contains the hypervolume indicator values and Tables (c) and (d) the epsilon ones. As a general remark, the configurations found by this proposal yield similar indicator values (each cell includes the median of 25 independent runs) than those presented in previous work [115]. Additionally, this results are also supported by the Wilcoxon rank sum statistical test for significance. The results of the Wilcoxon test are included in Table 5.4. Statistical confidence has been found in most of the results.

5.5 Discussion on the proposed experiments

After conducting the two defined experiments, the two research questions formulated in the introduction are revisited, and an attempt is made to answer them based on the results obtained.

5.5.1 Research Questions

This section includes the answers to the research questions formulated on the introduction.

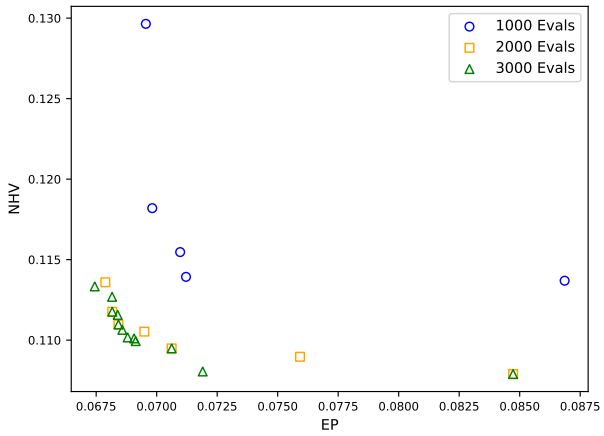


Figure 5.3: Problem DTLZ3. Evolution of the front generated by the meta-optimizer.

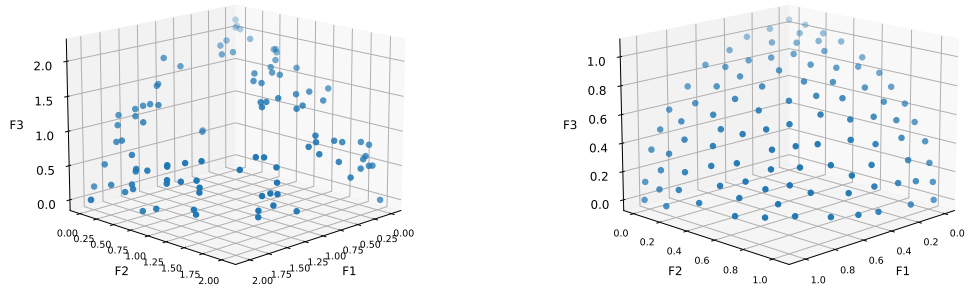


Figure 5.4: Problem DTLZ3. Pareto front approximation found by the standard NSGA-II (left), and Pareto front approximation found by the auto-designed NSGA-II (right).

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.35e-01_{1.8e-01}$	$1.17e-01_{8.0e-03}$	$6.34e-01_{1.8e-05}$
WFG2	$5.61e-01_{2.3e-03}$	$5.61e-01_{1.6e-03}$	$5.64e-01_{9.2e-05}$
WFG3	$4.92e-01_{8.3e-04}$	$4.92e-01_{6.1e-04}$	$4.95e-01_{6.1e-05}$
WFG4	$2.17e-01_{3.7e-04}$	$2.03e-01_{2.4e-03}$	$2.18e-01_{1.5e-03}$
WFG5	$1.95e-01_{3.5e-04}$	$1.96e-01_{7.8e-05}$	$1.96e-01_{9.8e-05}$
WFG6	$2.01e-01_{1.3e-02}$	$2.09e-01_{5.0e-04}$	$2.02e-01_{1.4e-02}$
WFG7	$2.09e-01_{5.5e-04}$	$2.09e-01_{2.7e-04}$	$2.11e-01_{3.5e-05}$
WFG8	$1.47e-01_{2.7e-03}$	$1.47e-01_{3.4e-03}$	$1.40e-01_{3.1e-03}$
WFG9	$2.37e-01_{1.8e-03}$	$2.35e-01_{5.8e-04}$	$2.39e-01_{2.0e-03}$
DTLZ1	$4.88e-01_{7.9e-03}$	$4.94e-01_{2.7e-04}$	$0.00e+00_{4.9e-01}$
DTLZ2	$2.09e-01_{4.7e-04}$	$2.10e-01_{1.6e-04}$	$2.11e-01_{3.6e-05}$
DTLZ3	$0.00e+00_{1.8e-02}$	$2.10e-01_{1.2e-01}$	$0.00e+00_{0.0e+00}$
DTLZ4	$2.09e-01_{2.1e-01}$	$2.10e-01_{8.7e-05}$	$2.11e-01_{7.2e-05}$
DTLZ5	$2.11e-01_{3.4e-04}$	$2.12e-01_{2.2e-04}$	$2.12e-01_{4.5e-05}$
DTLZ6	$1.82e-01_{3.6e-02}$	$2.12e-01_{8.1e-05}$	$2.12e-01_{4.2e-05}$
DTLZ7	$3.34e-01_{3.0e-04}$	$3.35e-01_{1.2e-04}$	$3.35e-01_{9.2e-05}$

(a) Current study results using as objective the hypervolume indicator.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.49e-01_{7.6e-02}$	$1.16e-01_{7.7e-03}$	$6.34e-01_{2.6e-05}$
WFG2	$5.64e-01_{9.5e-04}$	$5.62e-01_{1.2e-03}$	$5.65e-01_{5.1e-05}$
WFG3	$4.41e-01_{3.8e-04}$	$4.41e-01_{2.2e-04}$	$4.42e-01_{1.1e-05}$
WFG4	$2.17e-01_{7.6e-04}$	$2.03e-01_{2.4e-03}$	$2.17e-01_{3.0e-03}$
WFG5	$1.95e-01_{2.9e-04}$	$1.96e-01_{7.5e-05}$	$1.96e-01_{1.0e-04}$
WFG6	$2.03e-01_{8.9e-03}$	$2.09e-01_{4.3e-04}$	$2.08e-01_{1.3e-02}$
WFG7	$2.09e-01_{3.5e-04}$	$2.09e-01_{3.2e-04}$	$2.11e-01_{3.1e-05}$
WFG8	$1.48e-01_{2.3e-02}$	$1.48e-01_{1.0e-03}$	$1.39e-01_{2.3e-03}$
WFG9	$2.37e-01_{2.9e-03}$	$2.35e-01_{8.2e-04}$	$2.39e-01_{1.9e-03}$
DTLZ1	$4.66e-01_{1.6e-01}$	$4.94e-01_{1.9e-04}$	$0.00e+00_{0.0e+00}$
DTLZ2	$2.09e-01_{2.7e-04}$	$2.10e-01_{1.5e-04}$	$2.11e-01_{4.1e-05}$
DTLZ3	$0.00e+00_{0.0e+00}$	$2.10e-01_{6.3e-02}$	$0.00e+00_{0.0e+00}$
DTLZ4	$2.10e-01_{7.1e-04}$	$2.10e-01_{1.5e-04}$	$2.11e-01_{4.2e-05}$
DTLZ5	$2.11e-01_{3.5e-04}$	$2.12e-01_{1.3e-04}$	$2.12e-01_{4.1e-05}$
DTLZ6	$1.89e-05_{1.4e-03}$	$2.12e-01_{6.9e-05}$	$2.12e-01_{5.6e-05}$
DTLZ7	$3.29e-01_{2.8e-04}$	$3.30e-01_{9.8e-05}$	$3.30e-01_{7.3e-05}$

(b) Results obtained from [115] with irace using as objective the hypervolume indicator.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$2.94e-01_{2.7e-01}$	$4.56e-01_{1.3e-02}$	$6.19e-03_{5.6e-04}$
WFG2	$1.81e-01_{1.7e-01}$	$6.70e-03_{2.8e-03}$	$4.03e-03_{5.0e-04}$
WFG3	$1.33e-02_{2.8e-03}$	$7.39e-03_{8.4e-04}$	$5.40e-03_{2.2e-04}$
WFG4	$1.21e-02_{3.6e-03}$	$2.19e-02_{2.6e-03}$	$6.33e-03_{1.1e-03}$
WFG5	$3.31e-02_{2.8e-03}$	$2.78e-02_{3.9e-04}$	$2.76e-02_{1.8e-04}$
WFG6	$1.49e-02_{1.0e-02}$	$6.19e-03_{5.9e-04}$	$1.00e-02_{9.0e-03}$
WFG7	$1.20e-02_{4.4e-03}$	$6.34e-03_{9.1e-04}$	$5.18e-03_{2.6e-04}$
WFG8	$2.44e-01_{1.0e-01}$	$1.75e-01_{2.1e-02}$	$2.45e-01_{1.0e-03}$
WFG9	$1.47e-02_{2.7e-03}$	$1.12e-02_{1.2e-03}$	$7.19e-03_{1.6e-03}$
DTLZ1	$1.60e-02_{5.2e-03}$	$6.30e-03_{7.0e-04}$	$5.16e-01_{1.0e+00}$
DTLZ2	$1.23e-02_{2.8e-03}$	$5.59e-03_{3.7e-04}$	$5.23e-03_{2.2e-04}$
DTLZ3	$1.14e+00_{1.4e+00}$	$6.20e-03_{7.0e-01}$	$1.16e+01_{8.4e+00}$
DTLZ4	$1.18e-02_{9.9e-01}$	$5.68e-03_{3.7e-04}$	$5.38e-03_{3.0e-04}$
DTLZ5	$1.14e-02_{2.5e-03}$	$5.25e-03_{2.5e-04}$	$5.05e-03_{2.1e-04}$
DTLZ6	$2.77e-02_{2.4e-02}$	$5.22e-03_{6.6e-04}$	$5.09e-03_{1.9e-04}$
DTLZ7	$1.02e-02_{3.5e-03}$	$4.68e-03_{4.6e-04}$	$4.50e-03_{3.5e-04}$

(c) Current study results using as objective the epsilon indicator.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.52e-01_{2.4e-01}$	$4.55e-01_{9.8e-03}$	$5.99e-03_{7.3e-04}$
WFG2	$5.41e-03_{2.4e-03}$	$6.04e-03_{1.1e-03}$	$3.88e-03_{5.5e-04}$
WFG3	$3.34e-01_{5.3e-04}$	$3.34e-01_{1.9e-04}$	$3.33e-01_{2.7e-07}$
WFG4	$1.29e-02_{2.2e-03}$	$2.16e-02_{3.4e-03}$	$6.72e-03_{1.2e-03}$
WFG5	$3.31e-02_{3.7e-03}$	$2.77e-02_{1.9e-04}$	$2.76e-02_{1.6e-04}$
WFG6	$1.50e-02_{5.9e-03}$	$6.37e-03_{4.9e-04}$	$6.13e-03_{7.2e-03}$
WFG7	$1.28e-02_{4.0e-03}$	$6.52e-03_{4.9e-04}$	$5.20e-03_{1.9e-04}$
WFG8	$1.68e-01_{1.0e-01}$	$1.69e-01_{1.7e-02}$	$2.45e-01_{1.3e-03}$
WFG9	$1.42e-02_{2.0e-03}$	$1.10e-02_{1.9e-03}$	$7.39e-03_{1.2e-03}$
DTLZ1	$3.53e-02_{1.5e-01}$	$6.30e-03_{5.5e-04}$	$3.08e+01_{1.7e+01}$
DTLZ2	$1.12e-02_{4.2e-03}$	$5.53e-03_{3.2e-04}$	$5.28e-03_{2.5e-04}$
DTLZ3	$1.00e+01_{6.4e+00}$	$5.97e-03_{3.5e-01}$	$1.05e+02_{3.6e+01}$
DTLZ4	$1.18e-02_{5.4e-03}$	$5.61e-03_{3.1e-04}$	$5.32e-03_{1.9e-04}$
DTLZ5	$1.01e-02_{2.2e-03}$	$5.12e-03_{3.5e-04}$	$5.03e-03_{2.7e-04}$
DTLZ6	$3.72e-01_{5.3e-02}$	$5.15e-03_{4.5e-04}$	$5.06e-03_{2.9e-04}$
DTLZ7	$7.78e-03_{2.6e-03}$	$4.30e-03_{2.3e-04}$	$4.06e-03_{2.8e-04}$

(d) Results obtained from [115] with irace using as objective the epsilon indicator.

Table 5.3: The cells include the median and interquartile range of 25 independent runs. The dark-gray and light-gray background cells indicate, respectively, the best and second best indicator values.

(a) Current study results measuring the hypervolume indicator.

(b) Results obtained from [115] with irace measuring the hypervolume indicator.

(c) Current study results measuring the epsilon indicator.

(d) Results obtained from [115] with irace measuring the epsilon indicator.

Table 5.4: Wilcoxon rank sum test results. The symbols in each cell correspond to problems WFG1-9 and DTLZ1-7. The symbols indicate: "–" no statistical significance, "▲" the algorithm in the row has a better indicator value than the algorithm in the row with confidence and "▽" the algorithm in the row has a worse indicator value than the algorithm in the row with confidence.

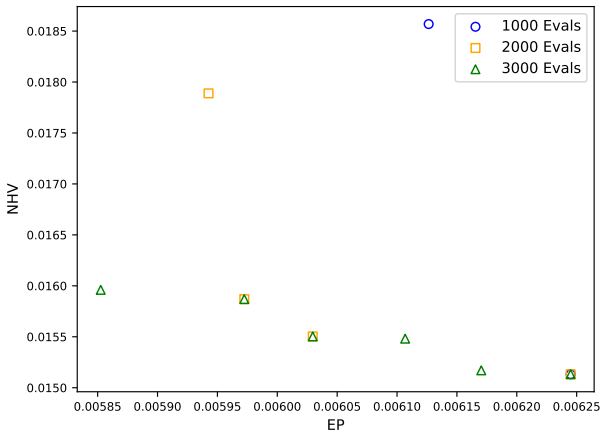


Figure 5.5: WFG problem family. Evolution of the front generated by the meta-optimizer.

RQ1 - approach complexity:

The proposed meta-optimization package relies only on jMetal code, so it does not require any external tool. The AutoNSGAII template within jMetal, designed and used in former studies, combined with irace simplifies the formulation of the auto-design of NSGA-II as a continuous optimization problem. Researchers familiar with jMetal are expected to benefit from the use of the meta-optimizer with little effort.

RQ2 - finding of accurate designs:

A simple encoding has been adopted for the NSGA-II configurations consisting in codifying each parameters as a floating point value in the range $[0.0, 1.0]$, and these values are further decoded into a string that used as the input of the AutoNSGAII template. This configuration has been proved effective by the obtained empirical experiments. The first two experiments showed that the meta-optimizer has been able to generate the expected key components required by NSGA-II to converge to the Pareto front of the selected problems. For these experiments visual evidence is provided. Additionally, the generalization capabilities of the auto-tuner were challenged by requiring it to find an accurate configuration for a training set composed of nine problems, and validating the found configurations on a set of additional seven problems. The experiment shows almost identical results to the previously published work were irace was used as auto-configuration tool.

5.5.2 Further Remarks

The provided empirical evaluation also revealed a few issues that are worth discussing:

- The formulation of searching designs for NSGA-II as a continuous problem opens the opportunity of using other metaheuristics provided by jMetal as meta-optimizers. This enable the easy development comparative studies based on configuring AutoNSGAII with different training sets.
- Although two quality indicators are used, EP and NHV, as objectives for guiding the search, the inclusion of additional ones (e.g., spread, inverted generational distance, etc.) could reveal new insights regarding the configurations for solving different problems.
- NSGA-II with standard settings is used as the meta-optimizer. The obtained results in this chapter could be used in order to analyze whether its performance could be improved if using different parameter settings.
- Only a run of the meta-optimizer has been performed in the experiments. A deeper study should be carried out by performing a number of independent runs and making statistical analysis of the results.

5.6 Implementing an auto-configuration tool

Following the validation of the proposal, the *Evolver* package was designed based on the jMetal-auto package, eliminating the need for external tools such as irace and thus simplifying the auto-design process for optimization problems implemented within that framework.

5.6.1 Software arquitecture

The software architecture of *Evolver* is represented in Figure 5.6. Its main components are:

- Configurable algorithm: This is a configurable template that implements a generic multi-objective metaheuristic. A specific algorithm can be instantiated by selecting a number of modular components. For instance, multi-objective evolutionary algorithms (MOEAs) are based on the generic template shown in Figure 2.3.
- Components and parameters: Components are modular blocks that provide useful utilities for building metaheuristics. Figure 2.3 shows some common

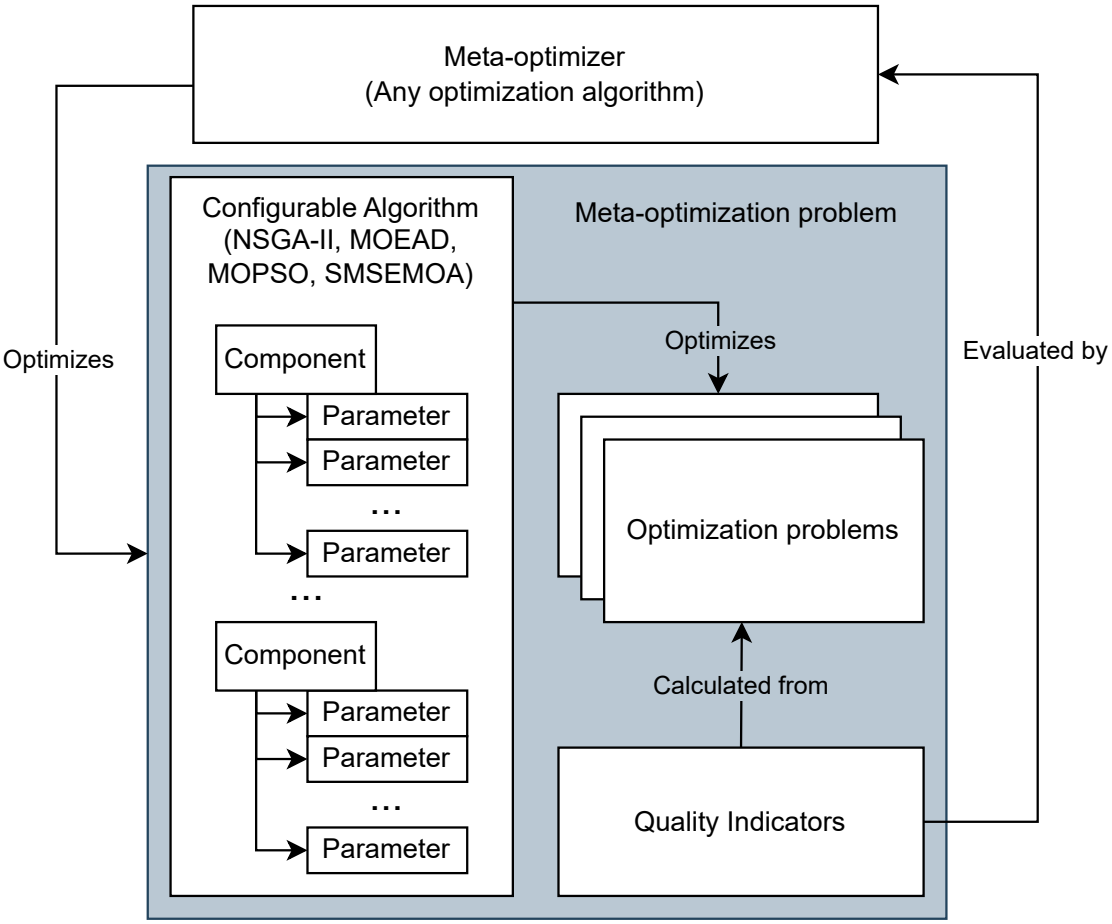


Figure 5.6: Software architecture of *Evolver*.

components used by MOEAs. Additionally, *Evolver* also implements more specific components some MOEAs are usually built upon, such as bounded or unbounded external solution archives. Parameters can be categorical, integer, or real. Examples of parameters include offspring population sizes, or probabilities for crossover and mutations operators (the latter two are the means by which MOEA generate new solutions). The set of all the available components and parameters and their relationships define the design space. As an example, Table 5.1 shows the design space of the configurable NSGA-II algorithm included in *Evolver*.

- Meta-optimization problem: Given a configurable algorithm and its corresponding design space, along with a set of problems to be optimized (i.e., the training set), the auto-configuration process is defined as a multi-objective optimization problem. The objective is to w.l.o.g. minimize a

number of quality indicators. As metaheuristics are stochastic-based methods, each algorithm configuration can be evaluated several times (i.e., a number of independent runs can be executed), being the objectives the median of the obtained indicator values. Real encoding is used to represent every parameter and component in a solution.

- Meta-optimization algorithms: These are conventional multi-objective algorithms for solving continuous problems. *Evolver* makes use of the algorithms provided by jMetal.

5.6.2 Software capabilities

Related work categorizes multi-objective metaheuristics into three categories: dominance-based, decomposition-based, and indicator-based. *Evolver* already includes the implementation of the most popular algorithm within each of these categories: NSGA-II (dominance-based), MOEA/D (decomposition-based), and SMS-EMOA (indicator-based). Additionally, it includes a configurable multi-objective particle swarm optimization algorithm (MOPSO) [40], which is another popular dominance-based variant.

Configurable algorithms have a greater number of components compared to their standard versions. For example, while the standard NSGA-II only includes a replacement strategy based on dominance ranking and a crowding distance density estimator, the implemented version by *Evolver* considers up to 19 different components (see Table A.1). The other configurable algorithms implemented follow the same pattern. MOEA/D, SMS-EMOA, and MOPSO configuration involves a set of 25, 18 and 24 components, respectively. Descriptions of these components for MOEA/D and MOPSO can be found in Tables A.3 and A.4 respectively, in the Appendix A. The parameter space of SMS-EMOA, Table A.2 is the same as NSGA-II except for the offspring population size. While NSGA-II it can take a value between 1 and 400, SMS-EMOA is a steady-state evolutionary algorithm (i.e., the value is always 1).

Evolver has a graphical user interface (GUI) in the form of a web application that enhances the configuration and execution of experiments. The main view of this GUI is illustrated in Figure 5.7. This interface offers several convenient functionalities to streamline the experimental process. Firstly, it provides easily modifiable fields with descriptive labels that enable users to configure the experiment according to their specific requirements. The GUI also offers real-time monitoring capabilities, enabling users to track the progress and status of an experiment as it executes. Additionally, the graphical user interface facilitates the management of experiment-related data by providing a convenient option to

download the artifacts and logs generated during the experiment.

For ease of use, both *Evolver* and the dashboard are also available as pre-built Docker images [107]. By encapsulating the software in containers, the need for manual setup and configuration is reduced, streamlining the deployment process. This approach ensures consistent and reliable performance across diverse environments while offering flexibility and adaptability for various deployment scenarios, from a user's laptop to cloud-oriented architectures.

5.7 Illustrative example

This section showcases how to use *Evolver* to auto-configure a metaheuristic for solving an engineering problem. This scenario resembles the typical role of an engineer interested in using NSGA-II to solve a specific class of engineering problem, but uncertain about how to configure it for the best outcome. With that goal in mind, the plan is to utilize an example that has a reference front available for training.

To simulate the mentioned scenario, the liquid-rocket single element injector design problem described in [167] has been selected. The problem is formulated with three objectives and four decision variables. As meta-optimizer, the engineer uses the GUI to select NSGA-II with the default settings, as shown in Figure 5.7 (population size = 50, maximum number of evaluations = 2000, independent runs = 1). The objectives to minimize are the inverted generational distance plus (IGD+) and the additive epsilon (EP).

The size of the population in the configurable NSGA-II is set by default to 100 (a common value). An important parameter to define is the stopping condition. The recommendation in [167] is to use as stopping condition $N \times 500$ function evaluations (N is the population size); however, here in this case, the condition is set to 7,000, with the intention of reducing the running time of *Evolver* and the hope that the found configuration can provide a NSGA-II variant able of successfully solve the target problem.

Figure 5.8 displays the configurations found with the best trade-off between IGD+ and EP across different numbers of evaluations. After *Evolver* finishes, let's assume the engineer selects the configuration with the minimum IGD+ and uses it to run NSGA-II for 20,000 problem evaluations. Figure 5.9 illustrates the reference front of the problem (top), the front obtained by NSGA-II with standard settings (bottom left), and the front found by the configurable NSGA-II (bottom right). From the figure, the NSGA-II mainly finds solutions at the

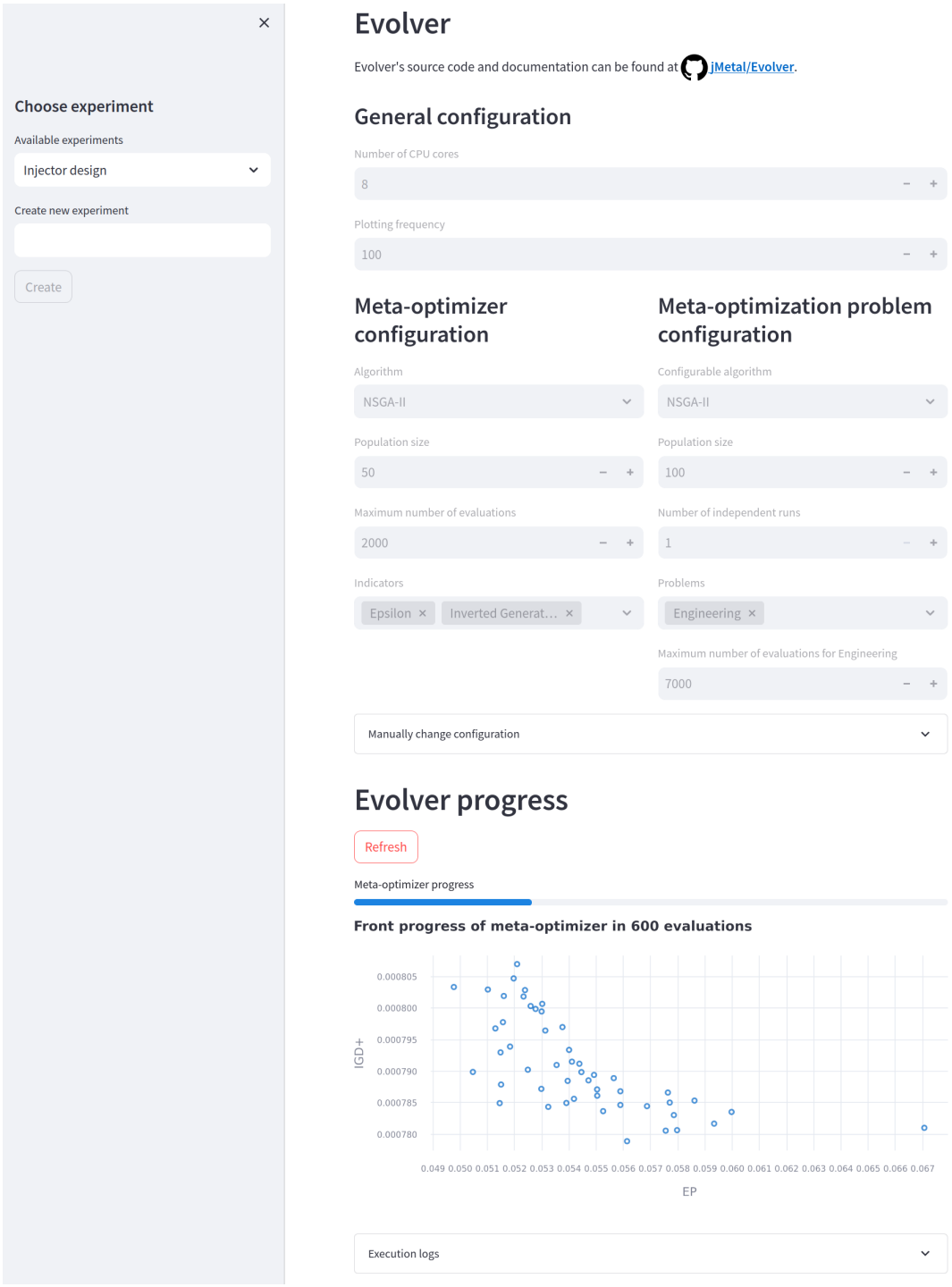


Figure 5.7: *Evolver*'s dashboard while executing the liquid-rocket single element injector design problem. The chart shows the population of the meta-optimizer after 600 function evaluations, which contains four non-dominated solutions.

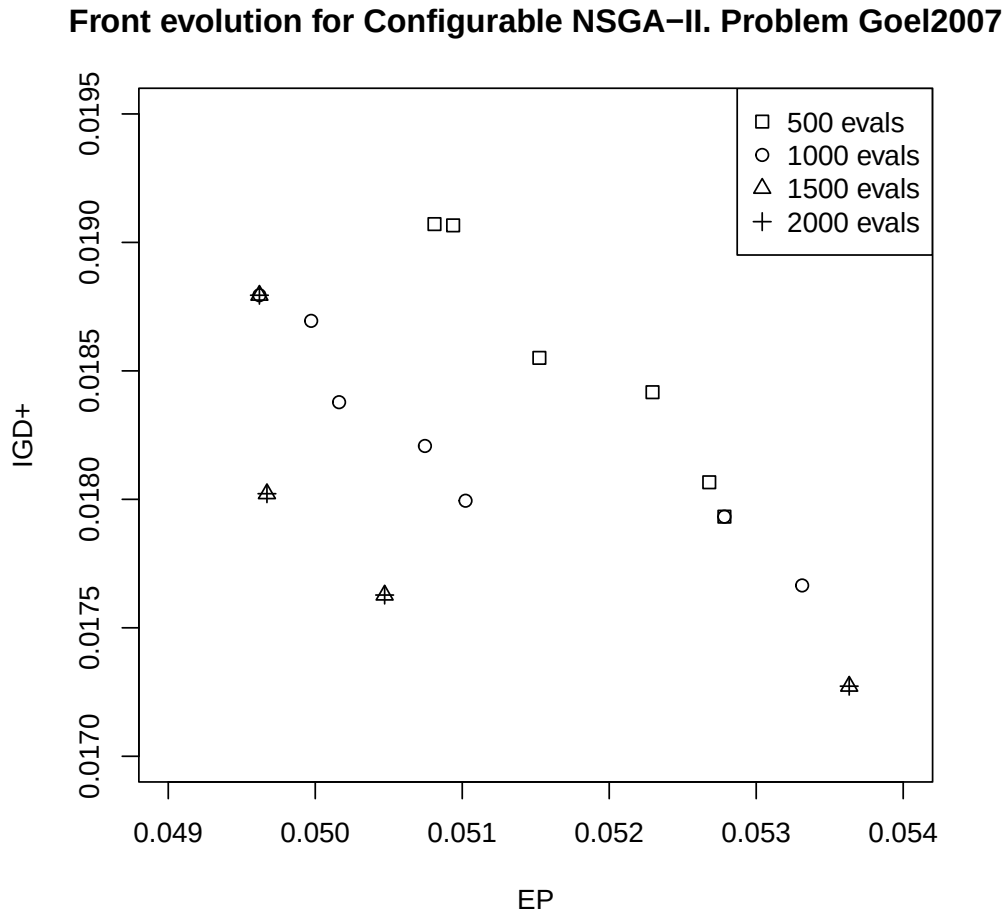


Figure 5.8: Evolution of the Pareto front approximations found by the NSGA-II meta-optimizer in the search for a configuration of NSGA-II for the engineering problem.

border of the reference front, while the auto-configured version produces a set of solutions uniformly distributed across the surface of the reference front.

5.8 Impact

Evolver offers a convenient and efficient solution for auto-configuring metaheuristics by providing a pure Java implementation, eliminating the need for external tools. Since *Evolver* uses jMetal as its main dependency, it should be easy to use for a wide community of researchers. The main papers describing jMetal, [41] and [113], currently have 1372 and 225 references, respectively, according to

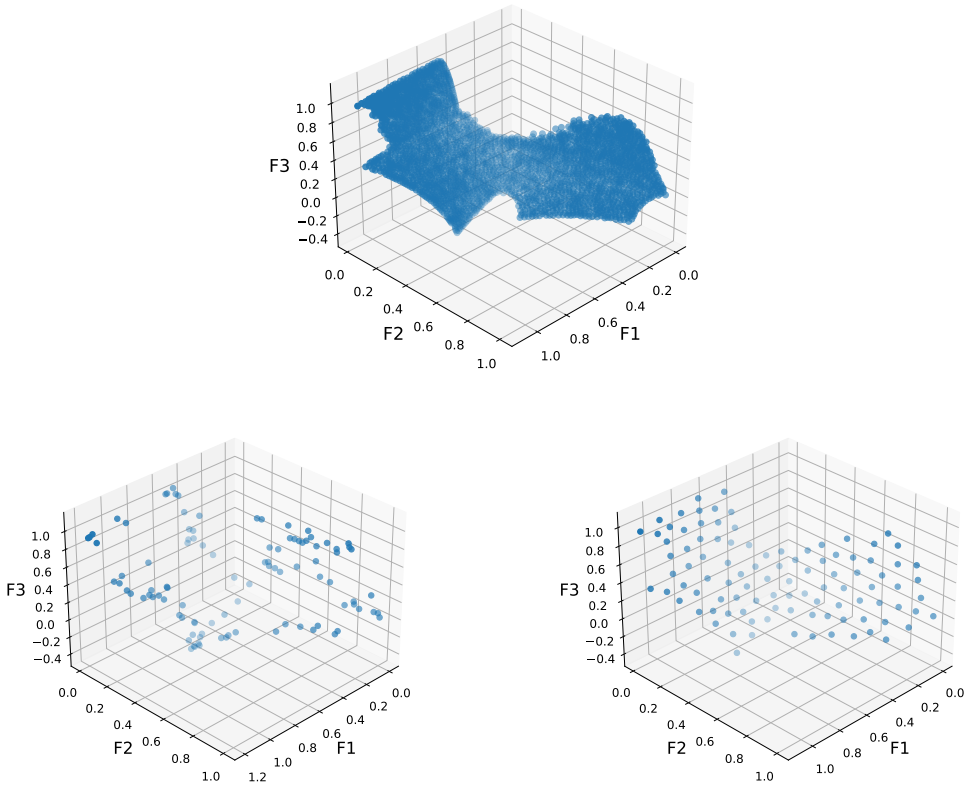


Figure 5.9: Reference front of the engineering problem (top), front obtained by NSGA-II with standard setting (bottom left), and front obtained by the auto-configured NSGA-II (bottom right).

Google Scholar [2].

For metaheuristics experts, *Evolver* is an invaluable tool not only for finding configurations and designing algorithms tailored to particular training sets, but also for researching automatic parameter tuning for metaheuristics. Some research lines that can be highlighted include:

- Finding suitable configurations for multi-objective metaheuristics is itself a multi-objective problem. Therefore, it can serve as a basis for designing benchmark suites that include various combinations of algorithms, quality indicators, and training sets. This approach would enable studies focused on searching for efficient meta-optimizers.
- Running a meta-optimization process can be a very computationally inten-

[2]Number of citations as of May 30th, 2024

sive task. Therefore, improving its efficiency is of great interest [73]. In this regard, it is worthwhile to investigate the minimum number of evaluations required by the configurable metaheuristic, given a training set. Such an investigation can help obtain effective algorithms that are capable of accurately obtaining Pareto fronts for the given problems.

- *Evolver* can generate a large amount of data that can be used to analyze the most influential parameters of the metaheuristics being tuned, as well as to address the search landscape characteristics of a given configuration problem.

For problem domain expert lacking experience in multi-objective metaheuristics, using *Evolver* through its GUI can facilitate the search of algorithm configurations once the target problem is defined in terms of the guidelines provided by jMetal. An example of this scenario is provided in the former section.

5.9 Quality-Diversity: An alternative approach for generating algorithmic configurations

This section aims to showcase an alternative approach to auto-configuration that has been studied during the course of this PhD. While this line of research is still open, this section describes the proposed approach as well as some preliminary results.

5.9.1 Introduction

In the field of machine learning, ensemble methods are a set of techniques that combine multiple models to improve the performance and robustness over individual models [124]. The main idea behind ensembles is that aggregating the outputs of several algorithms will provide the ensemble the strengths while mitigating the weaknesses of each constituent model, at the cost of higher computational cost. Additionally, it has been proven empirically that ensembles provide better results when there is greater diversity between the models that form it [142, 87].

When applied to multi-objective optimization, an ensemble of several optimization algorithms can be implemented by taking the resulting non-dominated populations of each of the constituent algorithms and merging them.

Quality-Diversity (QD) optimization is a branch of mono-objective stochastic optimization with the goal to move beyond single optimal solution and instead

exploring a wide range of high-quality solutions. These solutions apart from being high-quality with respect to the fitness function, they are diverse according a one or more user-defined features [126]. Each of these diversity features are called behavior characteristics, as they represent *how* a solution solves the problem. One of the most popular state-of-the-art algorithms for Quality-Diversity is the Covariance Matrix Adaptation MAP-Elites (CMA-ME) [49] that combines the search capabilities of Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [62] with the MAP-Elites [109] algorithm for maintaining diversity.

This section presents a novel approach for the auto-configuration of metaheuristics, where instead of searching for the best configuration to solve a set of problems, Quality-Diversity optimization is used to find a diverse set of quality configurations that can be used as an ensemble to improve the performance and generalization capabilities of the individual configurations found.

This section is structured as follows. First, the proposed approach is presented in Subsection 5.9.2, then in Subsection 5.9.3 further explores the specifics of the implementation provided. Finally, in Subsection 5.9.4 a preliminary evaluation is provided.

5.9.2 Quality-Diversity optimization for metaheuristics

To use Quality-Diversity for the automatic configuration of metaheuristics, first the optimization problem must be defined. To optimize an algorithm to a specific set of problems, later referred to as training set, the fitness function of the optimization problem will be refereed as the average normalized hypervolume (NHV). Like in *Evolver*, the search space of the optimization problem are the different components and parameters that can configure a specific metaheuristic. As the parameters can be both continuous or discrete, the optimization problem is of mixed-integer nature. From this point, this optimization problem will be refereed to as meta-problem, to avoid any confusion to the problems of the training set.

However, there are several challenges that need to be addressed. Firstly, the behavioral characteristics that will guide the search for diversity need to be defined. These behavioral characteristics must characterize the different ways to solve the problems of the training set. For this characterization, the evolution of the population generated by a specific configuration is monitored with the objective of finding configurations with different behaviors, such as early or late convergence. With this goal in mind, the selected behavioral characteristics are the NHV at fixed intervals during the evaluation process. In the evaluation provided in this work, the NHV at the percentiles 20, 40, 60 and 80 are used as

behavioral characteristics. As there are more than 3 behavioral characteristics, the best way to visualize the results is using a parallel coordinates plot, as shown in Figure 5.10.

Secondly, the CMA-ES algorithm struggles with mixed-integer problems, such as algorithmic configuration, as the smaller variance of the integer variables leads to stagnation of the optimization process that generates new solution candidates via a multivariate Gaussian distribution. To overcome this challenge, the CMA-ES with margins (CMA-ESwM) algorithm is used [60]. This algorithm adds a lower-bound to the marginal probabilities associated with the generation of integer variables inside the original CMA-ES algorithm.

5.9.3 Implementation details

To implement this approach, the *meta-qdo* Python module has been developed. *pyribs* [150] is a Python framework for the development of Quality-Diversity algorithm via composing a modular set of components. *meta-qdo* uses the *pyribs* module as the back-bone of the Quality-Diversity implementation for two main reasons. First, the modularity of its components allows the implementation of a new emitter to integrate the CMA-ESwM algorithm into the library. Secondly, *pyribs* delegates on the user the evaluation of the set of solutions allowing to be easily parallelized. This second benefit is used in *meta-qdo* by implementing an evaluator that allows for parallel and distributed execution of the evaluation of the population based on Dask [132], an open-source parallel computing library in Python that is designed to scale and optimize the performance of computations, enabling efficient use of multi-core processors and distributed systems.

meta-qdo provides a common interface to implement the meta-optimization problem and currently offers implementations for the jMetal and jMetalpy frameworks. In this work, the focus is on the Java-based version of jMetal as it provides more configurable algorithms.

For the evaluation of each instance, the evaluation of an instance is delegated to jMetal. For this reason, the search space is almost the same as the one used in *Evolver*, except for the removal of the parameters for the offspring population and archive population sizes. As mentioned in the previous subsection, the NHV at several fixed intervals measures the behavior of a specific instance. Due to how jMetal works internally and the use of the observer design patter to calculate at this fixed intervals, the offspring population size and the population size with archive parameters must be fixed to the same size as the population.

Finally, *meta-qdo* provides several visualization methods and a small dash-

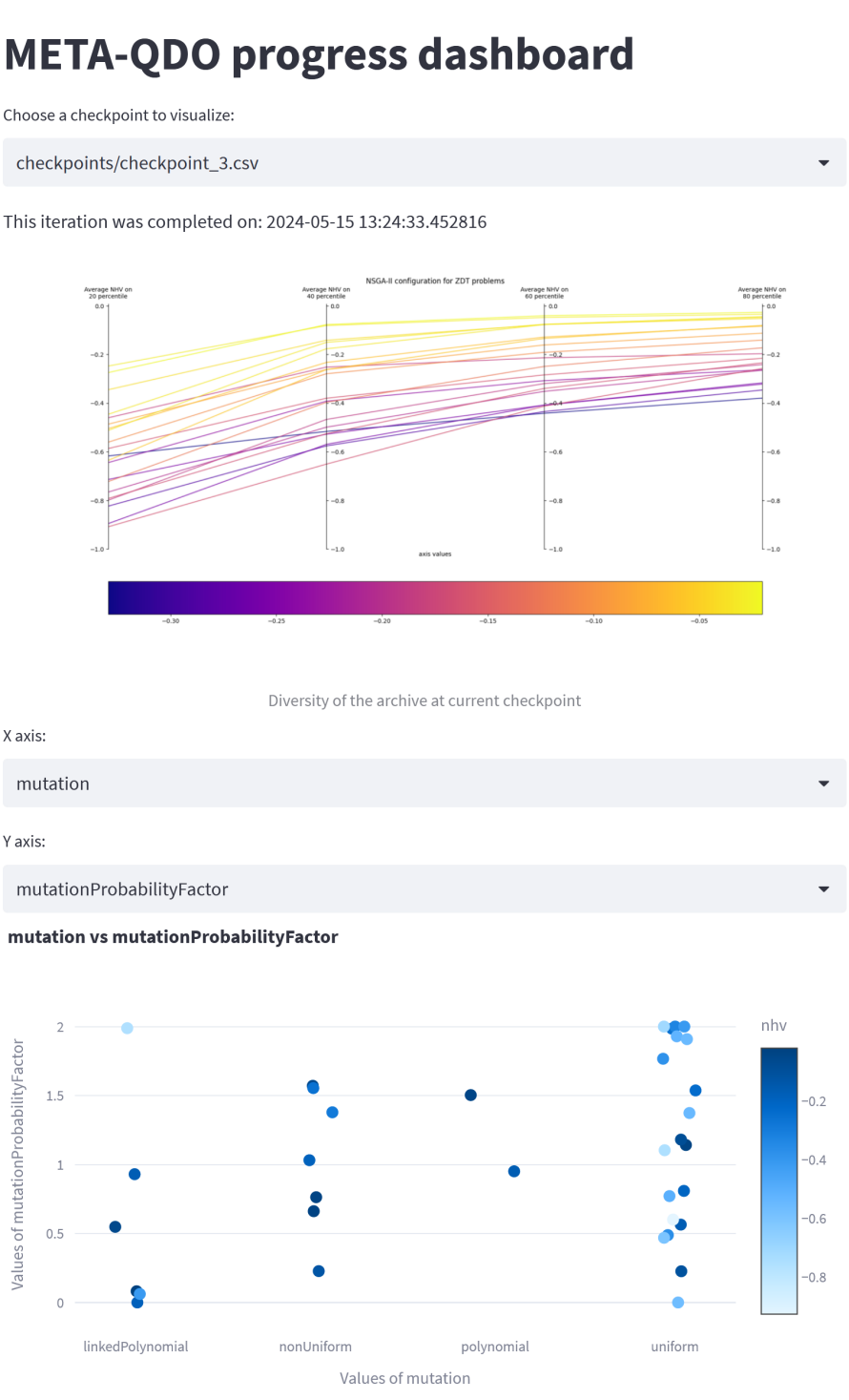


Figure 5.10: Dashboard for *meta-qdo* in the middle of an execution.

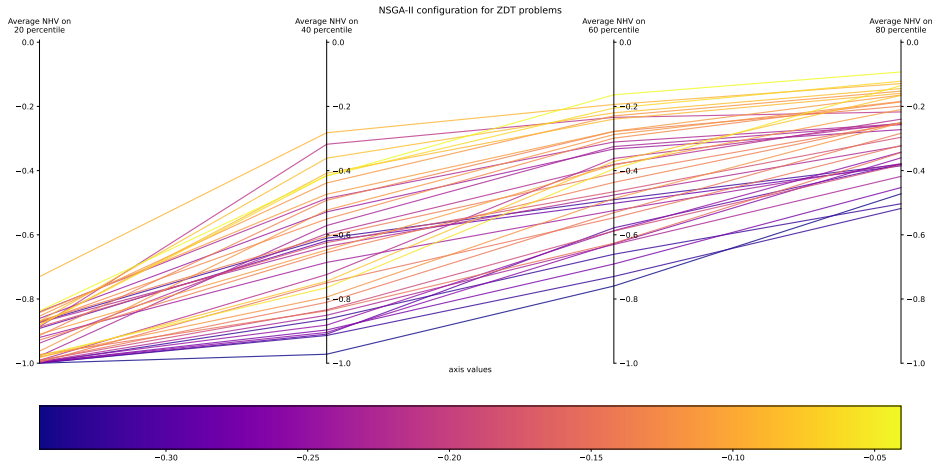


Figure 5.11: Parallel coordinates plot of the contents of a Quality-Diversity archive with 4 behavior characteristics measuring the progress of the NHV of the population of a multi-objective algorithm.

board for the monitoring of the progress of the current execution, allowing the user to visualize the diversity of the values of different parameters in relation to another. Figure 5.10 shows the execution of the optimization process from the dashboard.

5.9.4 Evaluation of the proposed approach

To validate the proposed approach, *meta-qdo* is run using the ZDT problems as the training set and later validated on the ZDT, WFG, DTLZ (with 2 objectives) and the 2-dimensional problems from the RE family. Once the Quality-Diversity algorithm has populated the archive, the top 10% of the elites according to their NHV values in the archive is selected. The different behaviors found by the Quality-Diversity algorithm can be seen in Figure 5.11, where visually several archetypes of behaviors can be observed, ranging from configurations that converge fast, and then stagnate, to the opposite.

Configuration	WFG1	WFG2	WFG3	WFG4
conf0	9.11e-02 ± 7.67e-02	2.17e-03 ± 7.14e-04	1.09e-02 ± 2.48e-03	3.34e-02 ± 7.31e-03
conf1	7.45e-01 ± 1.35e-02	7.35e-03 ± 3.22e-03	1.49e-02 ± 1.79e-03	2.26e-02 ± 1.83e-03
conf2	4.94e-01 ± 8.57e-02	7.77e-03 ± 2.67e-03	1.22e-02 ± 1.14e-03	2.98e-02 ± 5.42e-03
conf3	3.97e-01 ± 1.13e-01	7.86e-03 ± 1.78e-03	1.34e-02 ± 1.50e-03	2.03e-02 ± 2.06e-03
conf4	5.63e-01 ± 5.75e-02	6.57e-03 ± 2.67e-03	1.24e-02 ± 4.15e-03	3.28e-02 ± 5.47e-03
conf5	6.10e-01 ± 4.89e-02	5.83e-03 ± 2.24e-03	1.17e-02 ± 9.81e-04	3.11e-02 ± 5.03e-03
NSGA-II default	6.04e-01 ± 1.32e-01	5.77e-03 ± 2.14e-03	1.44e-02 ± 1.94e-03	2.48e-02 ± 2.14e-03
conf0-x6	1.16e-02 ± 7.04e-03	-4.94e-04 ± 1.94e-04	-1.36e-04 ± 2.82e-04	1.38e-02 ± 2.04e-03
conf1-x6	5.49e-01 ± 9.69e-02	1.33e-03 ± 7.96e-04	3.47e-03 ± 8.26e-04	4.63e-03 ± 7.51e-04
conf2-x6	2.91e-01 ± 8.68e-02	2.68e-03 ± 7.85e-04	3.57e-03 ± 6.09e-04	1.41e-02 ± 1.99e-03
conf3-x6	2.57e-01 ± 8.30e-02	3.32e-03 ± 8.24e-04	4.49e-03 ± 5.57e-04	6.98e-03 ± 9.00e-04
conf4-x6	4.84e-01 ± 5.21e-02	2.65e-03 ± 6.09e-04	3.08e-03 ± 8.58e-04	1.42e-02 ± 1.94e-03
conf5-x6	5.42e-01 ± 5.05e-02	3.07e-03 ± 4.12e-04	3.72e-03 ± 5.58e-04	1.30e-02 ± 2.05e-03
NSGA-II default-x6	3.75e-01 ± 9.13e-02	3.39e-04 ± 3.82e-04	3.79e-03 ± 8.52e-04	6.87e-03 ± 5.43e-04
conf-diverse-0-5	1.10e-01 ± 7.05e-02	9.15e-04 ± 6.20e-04	2.84e-03 ± 7.18e-04	9.74e-03 ± 1.01e-03
Configuration	**WFG5**	**WFG6**	**WFG7**	**WFG8**
conf0	1.66e-01 ± 6.04e-03	2.49e-02 ± 8.85e-03	2.33e-02 ± 1.70e-03	3.70e-01 ± 9.03e-03
conf1	1.68e-01 ± 1.52e-03	7.03e-02 ± 6.57e-02	2.60e-02 ± 1.86e-03	3.41e-01 ± 4.42e-03
conf2	1.62e-01 ± 1.60e-04	5.89e-02 ± 4.45e-02	2.39e-02 ± 1.87e-03	3.12e-01 ± 6.44e-02
conf3	1.56e-01 ± 1.31e-02	4.94e-02 ± 4.80e-02	2.44e-02 ± 2.11e-03	3.33e-01 ± 5.99e-03
conf4	1.59e-01 ± 7.04e-03	4.75e-02 ± 3.09e-02	2.19e-02 ± 1.75e-03	3.44e-01 ± 7.40e-03
conf5	1.62e-01 ± 2.23e-04	3.02e-02 ± 1.03e-02	2.32e-02 ± 1.38e-03	3.13e-01 ± 6.81e-02
NSGA-II default	1.68e-01 ± 1.94e-03	6.17e-02 ± 3.45e-02	2.72e-02 ± 2.16e-03	3.31e-01 ± 6.39e-03
conf0-x6	1.51e-01 ± 6.19e-04	5.77e-03 ± 3.21e-03	4.32e-03 ± 2.27e-04	3.22e-01 ± 3.34e-02
conf1-x6	1.52e-01 ± 1.36e-04	2.13e-02 ± 5.00e-03	7.87e-03 ± 6.12e-04	3.00e-01 ± 2.16e-02
conf2-x6	1.48e-01 ± 6.93e-03	8.57e-03 ± 2.58e-03	8.56e-03 ± 6.28e-04	2.13e-01 ± 7.62e-02
conf3-x6	1.35e-01 ± 1.86e-02	1.72e-02 ± 3.42e-03	1.03e-02 ± 9.45e-04	2.92e-01 ± 3.27e-02
conf4-x6	1.46e-01 ± 6.81e-03	9.46e-03 ± 3.81e-03	7.80e-03 ± 4.93e-04	2.34e-01 ± 7.48e-02
conf5-x6	1.49e-01 ± 6.45e-03	1.22e-02 ± 5.13e-03	8.67e-03 ± 5.88e-04	2.18e-01 ± 8.07e-02
NSGA-II default-x6	1.52e-01 ± 3.09e-04	2.22e-02 ± 4.98e-03	8.54e-03 ± 6.56e-04	2.60e-01 ± 4.77e-02
conf-diverse-0-5	1.42e-01 ± 1.08e-02	1.14e-02 ± 5.44e-03	7.85e-03 ± 3.40e-04	2.46e-01 ± 6.63e-02
Configuration	**WFG9**	**DTLZ1_2D**	**DTLZ2_2D**	**DTLZ3_2D**
conf0	4.58e-02 ± 1.05e-02	1.00e-00 ± 0.00e-00	2.32e-02 ± 1.74e-03	1.00e-00 ± 0.00e-00
conf1	4.00e-02 ± 4.96e-03	9.08e-01 ± 1.29e-01	2.88e-02 ± 2.13e-03	1.00e-00 ± 0.00e-00
conf2	4.06e-02 ± 1.33e-02	1.00e-00 ± 0.00e-00	2.89e-02 ± 1.92e-03	1.00e-00 ± 0.00e-00
conf3	3.85e-02 ± 4.71e-03	1.00e-00 ± 0.00e-00	4.73e-02 ± 6.04e-03	1.00e-00 ± 0.00e-00
conf4	3.52e-02 ± 5.37e-03	1.00e-00 ± 0.00e-00	2.87e-02 ± 2.72e-03	1.00e-00 ± 0.00e-00
conf5	4.06e-02 ± 4.90e-03	1.00e-00 ± 0.00e-00	2.90e-02 ± 1.58e-03	1.00e-00 ± 0.00e-00
NSGA-II default	4.12e-02 ± 5.26e-03	8.21e-01 ± 2.94e-01	2.62e-02 ± 1.58e-03	1.00e-00 ± 0.00e-00
conf0-x6	2.26e-02 ± 5.18e-03	9.98e-01 ± 4.59e-03	4.77e-03 ± 1.38e-04	1.00e-00 ± 0.00e-00
conf1-x6	1.99e-02 ± 2.23e-03	6.15e-01 ± 2.97e-01	9.23e-03 ± 4.28e-04	1.00e-00 ± 0.00e-00
conf2-x6	2.16e-02 ± 2.47e-03	7.12e-01 ± 3.82e-01	1.27e-02 ± 6.34e-04	1.00e-00 ± 0.00e-00
conf3-x6	2.13e-02 ± 1.12e-03	9.77e-01 ± 6.94e-02	2.40e-02 ± 2.74e-03	1.00e-00 ± 0.00e-00
conf4-x6	2.09e-02 ± 3.43e-03	8.93e-01 ± 2.23e-01	1.24e-02 ± 1.26e-03	1.00e-00 ± 0.00e-00
conf5-x6	2.09e-02 ± 1.76e-03	7.95e-01 ± 2.98e-01	1.35e-02 ± 7.15e-04	1.00e-00 ± 0.00e-00
NSGA-II default-x6	1.95e-02 ± 2.12e-03	4.85e-01 ± 2.51e-01	7.47e-03 ± 2.51e-04	1.00e-00 ± 0.00e-00
conf-diverse-0-5	1.98e-02 ± 2.51e-03	7.70e-01 ± 3.02e-01	1.05e-02 ± 6.93e-04	1.00e-00 ± 0.00e-00
Configuration	**DTLZ4_2D**	**DTLZ5_2D**	**DTLZ6_2D**	**DTLZ7_2D**
conf0	2.58e-02 ± 1.77e-03	1.69e-02 ± 1.34e-03	1.48e-02 ± 1.19e-03	1.38e-02 ± 1.34e-03
conf1	4.90e-02 ± 1.67e-02	2.06e-02 ± 1.44e-03	2.78e-02 ± 5.28e-03	2.37e-02 ± 6.73e-03
conf2	3.46e-02 ± 2.53e-03	2.29e-02 ± 1.99e-03	8.73e-03 ± 1.43e-04	1.12e-02 ± 1.47e-03
conf3	7.63e-02 ± 1.04e-02	3.99e-02 ± 7.37e-03	1.09e-02 ± 1.24e-03	1.84e-02 ± 4.22e-03
conf4	3.48e-02 ± 2.46e-03	2.35e-02 ± 3.76e-03	8.83e-03 ± 1.42e-04	1.09e-02 ± 1.74e-03
conf5	3.70e-02 ± 1.71e-03	2.37e-02 ± 1.90e-03	8.75e-03 ± 1.21e-04	1.17e-02 ± 2.70e-03
NSGA-II default	2.20e-01 ± 3.90e-01	1.96e-02 ± 1.66e-03	1.00e-00 ± 0.00e-00	4.86e-02 ± 8.00e-03

Table 5.5: Average normalized hypervolume for each configuration and ensemble when solving a specific problem. Highlighted in dark gray are the first best-performing algorithms, while those in light gray represent the second best performers.

Configuration	DTLZ4_2D	DTLZ5_2D	DTLZ6_2D	DTLZ7_2D
conf0-x6	7.18e-03 ± 4.54e-04	-1.66e-03 ± 2.33e-04	-3.56e-03 ± 1.70e-04	5.18e-03 ± 7.17e-04
conf1-x6	1.12e-02 ± 1.50e-03	3.36e-03 ± 1.44e-03	-2.13e-03 ± 3.03e-04	2.97e-03 ± 4.12e-04
conf2-x6	1.74e-02 ± 6.94e-04	6.63e-03 ± 6.58e-04	-4.05e-03 ± 1.99e-04	4.05e-03 ± 2.79e-04
conf3-x6	3.09e-02 ± 3.65e-03	1.89e-02 ± 2.33e-03	-4.01e-03 ± 1.70e-04	2.64e-03 ± 1.69e-04
conf4-x6	1.63e-02 ± 1.25e-03	6.11e-03 ± 8.21e-04	-4.07e-03 ± 1.94e-04	3.79e-03 ± 2.46e-04
conf5-x6	1.80e-02 ± 1.19e-03	7.16e-03 ± 7.11e-04	-3.98e-03 ± 3.06e-04	4.24e-03 ± 3.26e-04
NSGA-II default-x6	9.48e-03 ± 8.50e-04	8.98e-04 ± 2.09e-04	1.00e-00 ± 0.00e-00	2.80e-02 ± 1.86e-03
conf-diverse-0-5	1.40e-02 ± 5.03e-04	4.41e-03 ± 9.11e-04	-3.68e-03 ± 1.73e-04	3.48e-03 ± 2.75e-04
Configuration	**ZDT1**	**ZDT2**	**ZDT3**	**ZDT4**
conf0	1.20e-02 ± 9.09e-04	2.15e-02 ± 1.92e-03	2.07e-02 ± 4.88e-03	7.68e-01 ± 1.72e-01
conf1	1.49e-02 ± 2.02e-03	3.78e-02 ± 6.86e-03	1.20e-02 ± 2.17e-03	5.40e-01 ± 2.39e-01
conf2	9.60e-03 ± 7.20e-04	1.90e-02 ± 2.26e-03	1.40e-02 ± 3.41e-03	9.31e-01 ± 1.24e-01
conf3	1.19e-02 ± 1.65e-03	2.69e-02 ± 4.42e-03	8.89e-03 ± 2.42e-03	9.31e-01 ± 1.11e-01
conf4	9.88e-03 ± 6.83e-04	1.69e-02 ± 1.63e-03	1.27e-02 ± 2.16e-03	9.57e-01 ± 6.68e-02
conf5	9.85e-03 ± 1.11e-03	1.91e-02 ± 1.91e-03	1.60e-02 ± 4.64e-03	9.85e-01 ± 3.22e-02
NSGA-II default	4.14e-02 ± 4.28e-03	3.82e-01 ± 2.79e-01	4.41e-02 ± 9.23e-03	8.08e-01 ± 1.48e-01
conf0-x6	4.19e-03 ± 2.63e-04	8.31e-03 ± 7.26e-04	9.81e-03 ± 1.63e-03	4.90e-01 ± 2.43e-01
conf1-x6	2.73e-03 ± 1.17e-04	6.58e-03 ± 2.57e-04	1.72e-03 ± 1.68e-04	3.13e-01 ± 8.80e-02
conf2-x6	3.83e-03 ± 3.89e-04	6.97e-03 ± 5.21e-04	6.80e-03 ± 1.43e-03	7.38e-01 ± 1.32e-01
conf3-x6	2.63e-03 ± 9.64e-05	5.92e-03 ± 1.82e-04	1.59e-03 ± 1.53e-04	8.41e-01 ± 1.34e-01
conf4-x6	3.31e-03 ± 2.22e-04	6.43e-03 ± 4.06e-04	5.81e-03 ± 1.48e-03	7.02e-01 ± 2.41e-01
conf5-x6	3.95e-03 ± 3.90e-04	7.43e-03 ± 5.80e-04	7.64e-03 ± 1.03e-03	8.40e-01 ± 1.18e-01
NSGA-II default-x6	2.95e-02 ± 1.86e-03	8.94e-02 ± 9.95e-03	3.02e-02 ± 3.17e-03	4.71e-01 ± 2.02e-01
conf-diverse-0-5	3.33e-03 ± 2.00e-04	6.51e-03 ± 3.15e-04	3.64e-03 ± 8.48e-04	6.65e-01 ± 1.94e-01
Configuration	**ZDT6**	**RE21**	**RE22**	**RE23**
conf0	1.67e-02 ± 8.20e-04	1.20e-02 ± 4.90e-04	1.52e-02 ± 7.09e-04	1.37e-03 ± 1.73e-04
conf1	3.83e-02 ± 9.41e-03	1.21e-02 ± 6.88e-04	5.08e-01 ± 4.92e-01	8.01e-03 ± 1.65e-02
conf2	1.17e-02 ± 3.89e-04	7.66e-03 ± 2.25e-04	1.01e-02 ± 3.39e-04	7.89e-04 ± 3.15e-05
conf3	2.64e-02 ± 8.26e-03	7.43e-03 ± 1.64e-04	1.87e-02 ± 2.19e-02	1.27e-03 ± 2.89e-04
conf4	1.16e-02 ± 4.87e-04	7.68e-03 ± 1.82e-04	1.48e-02 ± 1.43e-02	8.15e-04 ± 4.78e-05
conf5	1.16e-02 ± 4.20e-04	7.55e-03 ± 1.62e-04	9.89e-03 ± 3.48e-04	8.30e-04 ± 5.99e-05
NSGA-II default	6.03e-01 ± 5.64e-02	1.16e-02 ± 4.57e-04	1.59e-02 ± 1.22e-03	1.17e-03 ± 1.42e-04
conf0-x6	3.01e-03 ± 1.73e-04	3.64e-03 ± 1.05e-04	4.69e-03 ± 4.03e-04	3.75e-04 ± 3.11e-05
conf1-x6	1.50e-02 ± 1.49e-03	3.82e-03 ± 1.23e-04	6.51e-03 ± 1.02e-03	5.80e-04 ± 1.33e-04
conf2-x6	2.41e-03 ± 1.55e-04	2.36e-03 ± 1.26e-04	3.15e-03 ± 1.79e-04	2.42e-04 ± 2.14e-05
conf3-x6	4.39e-03 ± 3.63e-04	2.25e-03 ± 7.41e-05	3.16e-03 ± 4.31e-04	3.67e-04 ± 5.18e-05
conf4-x6	2.45e-03 ± 1.80e-04	2.34e-03 ± 9.12e-05	3.13e-03 ± 1.76e-04	2.53e-04 ± 2.52e-05
conf5-x6	2.42e-03 ± 1.35e-04	2.38e-03 ± 9.82e-05	3.15e-03 ± 2.02e-04	2.52e-04 ± 3.28e-05
NSGA-II default-x6	5.00e-01 ± 2.23e-02	3.53e-03 ± 8.87e-05	5.07e-03 ± 2.33e-04	3.24e-04 ± 1.93e-05
conf-diverse-0-5	3.42e-03 ± 2.58e-04	2.60e-03 ± 6.40e-05	3.56e-03 ± 3.08e-04	2.95e-04 ± 2.19e-05

Configuration	RE24	RE25
conf0	2.02e-03 ± 1.31e-04	7.60e-10 ± 1.10e-10
conf1	2.34e-03 ± 1.03e-04	4.08e-09 ± 3.88e-09
conf2	1.47e-03 ± 3.57e-05	9.40e-10 ± 2.70e-10
conf3	1.37e-03 ± 4.08e-05	1.72e-09 ± 1.09e-09
conf4	1.45e-03 ± 6.03e-05	1.14e-09 ± 9.20e-10
conf5	1.47e-03 ± 5.22e-05	1.26e-09 ± 7.20e-10
NSGA-II default	2.35e-03 ± 2.88e-04	7.10e-10 ± 1.10e-10
conf0-x6	4.47e-04 ± 3.65e-05	3.00e-10 ± 2.00e-11
conf1-x6	5.59e-04 ± 3.04e-05	3.60e-10 ± 1.20e-10
conf2-x6	3.21e-04 ± 3.22e-05	3.30e-10 ± 7.00e-11
conf3-x6	2.86e-04 ± 2.72e-05	4.20e-10 ± 2.00e-10
conf4-x6	3.07e-04 ± 2.28e-05	3.30e-10 ± 7.00e-11
conf5-x6	3.31e-04 ± 3.25e-05	3.00e-10 ± 6.00e-11
NSGA-II default-x6	6.33e-04 ± 3.87e-05	2.90e-10 ± 1.00e-11
conf-diverse-0-5	3.58e-04 ± 2.31e-05	3.30e-10 ± 7.00e-11

Table 5.6: Continuation of average normalized hypervolume for each configuration and ensemble when solving a specific problem. Highlighted in dark gray are the first best-performing algorithms, while those in light gray represent the second best performers.

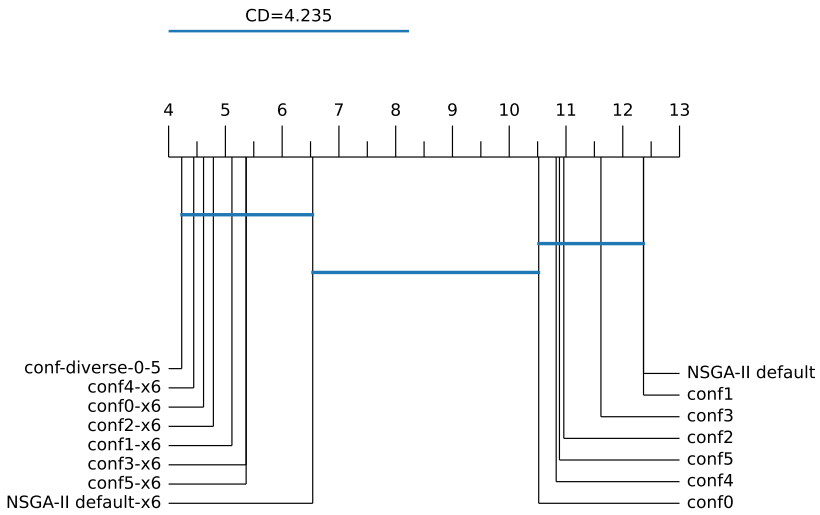


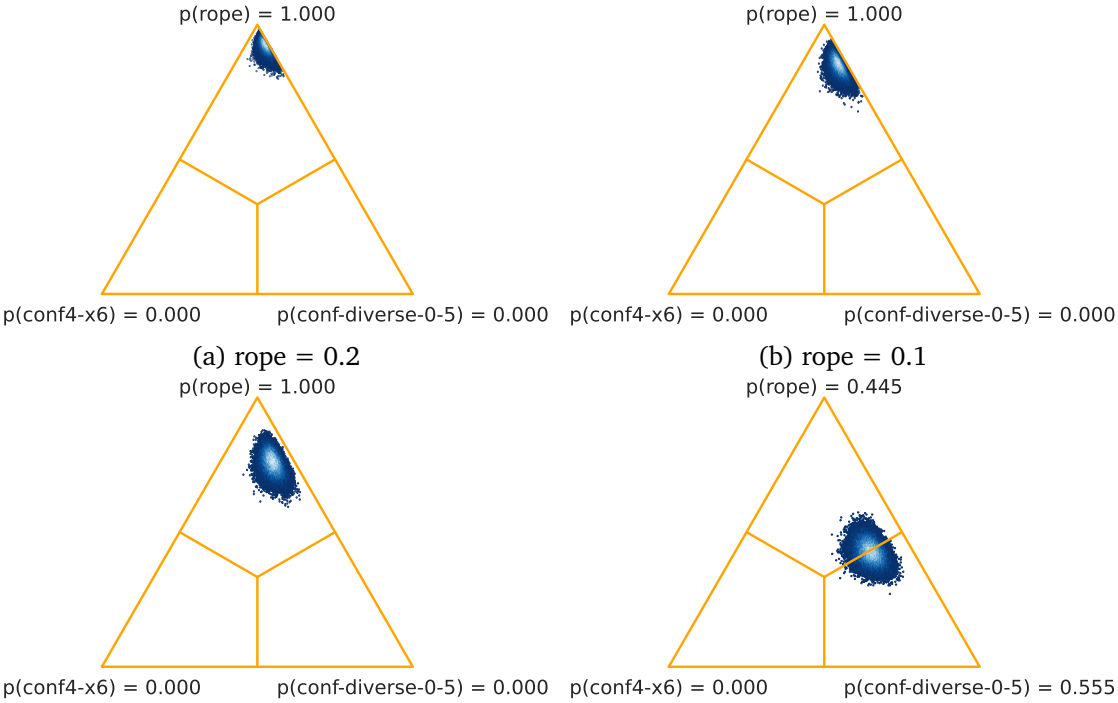
Figure 5.12: Critical distance plot ranking the obtain configurations and ensembles from the Quality-Diversity optimization process.

Then, an ensemble of the diverse configurations is implemented inside jMetal by solving each problem with each configuration and then merging the fronts to obtain a new front of non-dominated solutions. However, it would not be fair to compare a ensemble with n-times [3] the computational budget of a single configuration when solving the problem. For fairness, the diverse ensemble is compared with ensembled formed by n-times each of the configurations, including the default NSGA-II.

Tables 5.5 and 5.6 show the mean NHV with a budget of 10000 evaluations for 10 independent executions of each configuration on the validation problems. Ensembles have a budget of 10000 for each of their configurations.

To rank the obtained configurations, the critical distance (CD) plot is used over the average ranking of each of the algorithms computed over the considered problems [38]. The chart connects with a blue line those configurations whose difference in ranks is smaller than the critical distance. The critical distance is a function of the number of problems and configurations under comparison, as well as a critical value that results from the Studentized range statistic and a specified confidence level. Figure 5.12 shows the ranking of both the obtain con-

[3] Being n the number of unique configurations.



(a) rope = 0.2 (b) rope = 0.1

(c) rope = 0.01 (d) rope = 0.001

Figure 5.13: Posterior plot of the normalized hypervolume indicator using a Bayesian sign test for the top two performer ensembles with several different rope values.

figurations and the ensembles generated from them. The diverse ensemble is ranked first, but it has shown to be statistically equivalent according to the critical distance computed. The equivalence is due to the relatively higher strictness of the critical distance, particularly in benchmark problems where improvements are typically marginal.

To further study the difference between the top 2 ranked ensembles, a Bayesian sign test analysis [8] is provided in the Posterior plot shown in Figure 5.13. The Bayesian sign test defines a region of practical equivalence, also known as rope, that accounts for ties between the ranking of the algorithms, while the other two choices represent whether one of the configurations is superior to the other or vice versa. Figure 5.13 shows the analysis with different values of rope, the parameter that dictates the region of equivalence between the two configurations, which is defined in terms of the absolute difference of the paired NHV values between the two configurations. The probability of each of three possible outcomes can be estimated by counting the number of data points that fall in each of the regions. With this data and using 0.001 as the value of rope, the probability of the diverse ensemble being superior over the defined set of problems to

the second top-performer configuration is around 55.7%, while being practically equivalent in the rest of cases, proving that generating is never statistically worse than even the best configuration found.

This section has introduced a novel approach for the automatic configuration of metaheuristics by applying Quality-Diversity techniques to generate a diverse set of high-quality algorithmic configurations for forming an ensemble. This ensemble was evaluated on a set of benchmark problems to validate the proposed approach. A future step will be the evaluation on real-world problems.

Chapter 6

Leveraging Large Language Models for the Automatic Implementation of Optimization Problems

This chapter proposes the use of large language models for the automatic implementation of multi-objective optimization problems, allowing domain experts to implement their problems in optimization frameworks. For this, *moostral*, a fine-tuned version of a large language model, is presented to solve this task and embedded into a graphical tool to facilitate its usage and validate the provided implementation. The graphical tool provided by *moostral* lowers the implementation skill required to benefit from the recommendation system and other tools presented in this PhD thesis.

6.1 Introduction

Domain experts frequently deal with multi-objective optimization problems in real-world scenarios and metaheuristic techniques have become a popular alternative to solve them [28, 32]. One of the main reasons for the popularity of multi-objective metaheuristics stems from the existence of software packages like jMetal [41], PlatEMO [148] or Pymoo [15] that include the implementation of state-of-the-art algorithms, benchmark problems and utilities (e.g., quality indicators). Additionally, many of these packages offer a platform to develop custom techniques and problems that can readily benefit from all the package components.

However, there is a burden for many domain experts who, despite their deep understanding of the problem at hand, frequently lack the programming expertise in a specific programming language or are simply not familiar with the requirements imposed by the software engineering principles applied in the development of the optimization frameworks. For these users, software complexity may hinder the exploitation of these software packages.

Large Language Models (LLMs) have recently revolutionized the field of Artificial Intelligence and become the *de facto* approach for tasks related to the understanding of human language [171]. In particular, generative LLMs deal with the task of generating outputs that resemble the used training data. For example, a generative LLM trained with a large corpus of data can generate text that follows the same structure and grammar rules as the training data text. Interestingly, LLMs are not only restricted to human languages but have also shown success in generating machine language (e.g., programming languages). This fact has been exploited by tools such as GitHub Copilot[1] that support users by providing programming suggestions, for example.

LLMs, however, pose challenges in high consumption of computation resources, memory, and energy. Therefore, executing them is often beyond the resource capabilities of single users. LLMs as a service has become a standard mode of operation today, where users can use LLMs that run on a large pool of resources via API calls. While this solves the resources issue, it entails a financial cost, therefore impeding its usage for many users, and might also have environmental consequences [131].

In this PhD thesis, LLMs are used to bridge the gap between domain experts and multi-objective software packages, facilitating non-technical users to implement their multi-objective optimization problems, thus lowering the barrier of

[1] https://github.com/features/copilot

entry to benefit from their ecosystem. This chapter investigates one approach to address this issue is by employing Mistral [80] as the LLM and jMetal [41] as the target optimization framework. Additionally, the goal is to research this hypothesis while keeping the computational resources needed within the capabilities of single users.

The main contribution of this work is fine-tuning the smallest Mistral model to automatically implement multi-objective optimization problems in jMetal from a textual representation of the formulation. The model is trained by using synthetic multi-objective problems as inputs and the weights after fine-tuning are released under an open license. These synthetic problems are created by a problem generator also designed in the context of this work. Additionally, to facilitate its usage, the model is packaged inside *moostral*, a tool that implements validation steps to guarantee the correctness of the implementation and features a graphical user interface (GUI), all without requiring enterprise-level hardware.

The use of LLMs in the context of optimization with metaheuristics is gaining attention. A recent survey about evolutionary computation and LLMs [162] shows two main areas of interest: the use of LLMs as black-box search operators, and the utilization of the representation and generation abilities of LLMs to select or generate suitable optimization algorithms for solving specific problems. Some works related to LLM-based approaches to design metaheuristics include [101] and [98]. In the context of multi-objective optimization, Liu et al. investigates in [97] the application of LLMs to design operators for the MOEA/D multi-objective evolutionary algorithm. None of these works address the approach of using LLMs to automatically generate the implementation of multi-objective problems.

The rest of this chapter is structured as follows: In Section 6.2, the challenge of generating a synthetic dataset of multi-objective problems is approached. Section 6.3 details the process of fine-tuning an LLM for the automatic implementation of multi-objective optimization problems and explains the validation process utilized to guarantee the correctness of said implementation. Finally, Section 6.4, the model is evaluated on a set of real-world problems and the natural language definitions of problems.

6.2 Synthetic Problem Generation

One of the biggest challenges in machine learning is having a quality and sizable task-specific dataset. In the context of this work, this dataset would consist of pairs $(input, output)$, where $input$ is the textual description of a multi-objective

optimization problem and $output$ is the class implementing it. In jMetal, $Problem$ is a generic interface that requires the user to implement methods for gathering the number of decision variables, objectives and constraints. It also requires implementing methods to create a feasible solution for that problem and allow evaluating the objective functions for a specific solution. This work focuses on continuous optimization, and without loss of generality, all objective functions comprising the problems are assumed to be minimized. jMetal requires that classes implementing continuous problems extends the *DoubleProblem* interface (which in turn inherits from *Problem*), where methods for getting upper and lower bounds of each of the problem decision variables must be included.

Unfortunately, the number of problems currently implemented within jMetal (or any other framework) is not large enough to train a machine learning model. To overcome the dataset size challenge, a novel synthetic problem generator is developed. Some approaches for automatic problem generation exist in the related work using an affine combination of existing functions [39] or the random composition of trees from a known set of unary, binary, and vector-oriented operators operators [149]. The idea of this work is to artificially create valid pairs that could be used for fine-tuning a pre-trained model, but unfortunately these existing proposals from the literature did not contain the textual representation of a problem formulation nor the respective jMetal implementation. The proposed generator provides both. It utilizes LLMs and prompt engineering techniques to generate new multi-objective optimization problems. This methodology for generating synthetic data is valid due to the mathematical nature of optimization problems, which allows for verifying the correctness of the generated output for each input.

The problems created with the generator are not expected to be of interest from the point of view of optimization, e.g., multi-objective problems having Pareto sets with specific properties. The goal is to produce a sufficiently large dataset of diverse and valid problems that can be used to fine-tune a pre-trained LLM.

This generator also leverages LLMs, in particular a variant of OpenAI's GPT-3 [22] labeled as GPT-3.5 Turbo[2]. It follows a three-steps pipeline to generate multi-objective optimization problems. To improve the quality of the results, few-shot learning was utilized. Figure 6.1 provides a visual overview of the pipeline; a more in-depth description of each step is provided in the following.

The first step consists of a couple of functions that build a textual representation of the formulation of a multi-objective problem and the value ranges for

[2] https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates

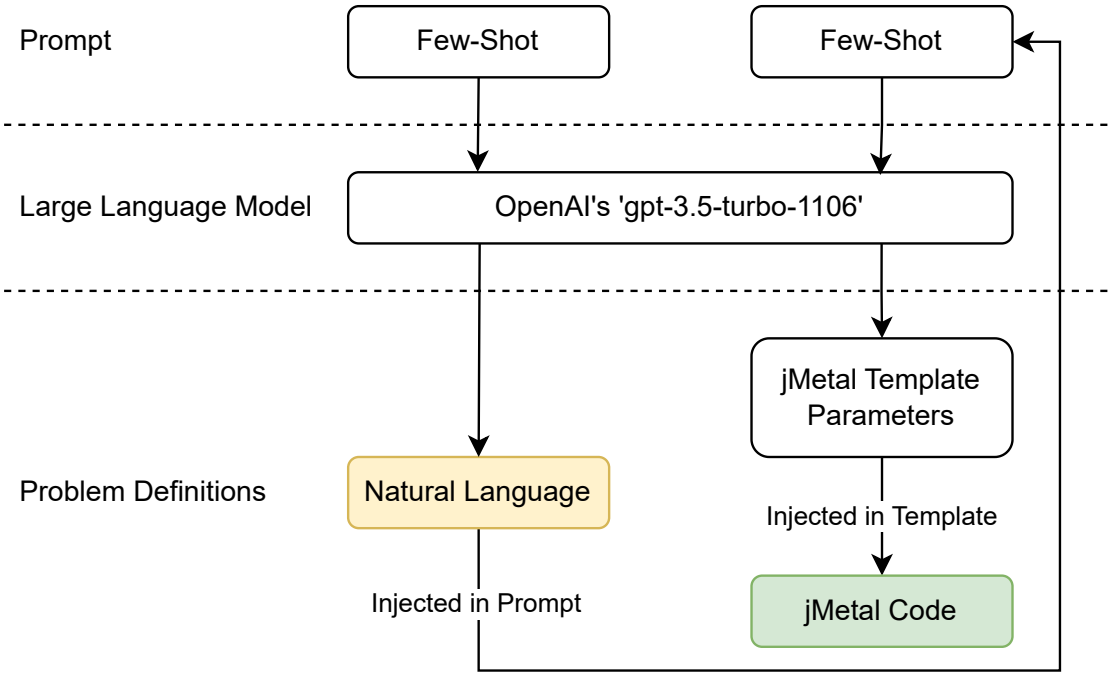


Figure 6.1: Pipeline for the generation of a synthetic dataset of multi-objective optimization problems.

its decision variables. This description is referred to as a prompt. To guarantee the diversity of the problems generated, a series of parameters with randomly chosen values are injected into the prompt. Table 6.1 showcases the parameters and possible choices for each of them.

The next step consists in generating the Java class implementing the problem in jMetal out of the prompt. This stage poses some additional challenges: the class must compile and allow running any algorithm included in jMetal. Initially, the model was tasked to generate the whole code of the class. However, it was observed that GPT-3.5 Turbo struggled when doing so and, in most cases, generated incomplete code. As the purpose of the generator was to create a dataset including valid implementation of problems, this task was simplified by asking the model to generate only several parameters and functions implementation out of the prompt. These parameters and functions are later rendered into a template for the implementation of the jMetal problem interface.

The template instantiated with the parameters and functions generated with the model is then compiled and validated as part of the last step. The validation

Parameter	Possible values
Number of objectives	[2, 5]
Number of variables	[2, 7]
Function type	Linear, quadratic, polynomial, rational, exponential, logarithmic, trigonometric or hyperbolic

Table 6.1: Parameters of the prompt for the generation of a new synthetic problem. The number of function types provided in the prompt are equal to the number of objectives.

consists of performing a battery of unit tests intended to ensure code correction. Upon the event of a failing test, that prompt and class are discarded. The tests performed in this stage do not guarantee a flawless implementation of the problem described in the formulation from the input; however, as this is only a training dataset, some small errors will not affect the training of the model. The goal is for the model to learn the problem structure and patterns within jMetal code; a deeper validation is provided on the developed tool described in Section 6.3.2.

SyntheticAI, the synthetic problem generator, is implemented in Python using OpenAI's library for communicating with the GPT-3.5 Turbo through their API. The problem generator is provided under an open-source license[3].

6.3 Methodology

Next, the methodology followed for designing the automatic implementation tool is described. First, the focus is directed on the selected LLM and how it was adapted to the proposed scenario. Second, the validation process of the outputs generated by the tool is described. The performance of the tool will be evaluated in the next section.

6.3.1 Fine-Tuning Mistral for Automatic Problem Implementation

Once a high-quality dataset is generated, it is used to adapt an LLM to perform better for the task of automatic multi-objective problem generation. More specifically, this process consists of selecting a foundational model and fine-tuning it with the generated dataset via self-supervised learning using LoRA. A dataset of

[3]Problem generator available at: `https://gitlab.com/jfaldanam-phd/syntheticai`

950 synthetic problems is used for this task. The fine-tuning process relies on the Huggin Face libraries *transformers* [159] for training and inference, and the *Parameter-Efficient Fine-Tuning* (PEFT) [164] for the LoRA implementation.

As the foundational model, Mistral-7B-v0.1 has been selected due to its small number of parameters (7 billion, which can be considered as small in the context of the state-of-the-art LLMs), its open license (Apache 2.0 license) and its high performance compared to other open models above its size. Mistral 7B has been shown to outperform higher parameters models [80] such as Llama 2 13B [152] across all evaluated benchmarks and Llama 1 34B [151] in reasoning, mathematics, and code generation.

The considered version of Mistral used in this paper encodes each of its parameters with two bytes (i.e., fp16). Storing the whole model into volatile memory requires around 24 GB of capacity without performing any optimization. This resource usage is within the capacity of powerful laptop or desktop machines. Related work shows that by applying techniques such as quantization it is possible to easily reduce the memory requirements by 2x and 4x [54], or even beyond [104] while retaining most of its language capabilities, and tools to perform quantization at the time of loading the model to do inference are available from different sources (e.g., Nvidia TensorRT[4] or the same Transformer library used in this work).

The model weights for *moostral-7B*, obtained after fine-tuning, are released under an open source license[5], along with *moostral*, the tool described in this work[6].

6.3.2 Guaranteeing Correctness While Working With Large Language Models

The main target of this work are domain experts who want to use an optimization framework like jMetal but lack the expertise to code their optimization problems by themselves. The proposed tool automatizes this task by providing an implementation out of a human description through a graphical user interface, depicted in Figure 6.2.

LLMs are probabilistic methods, and it is well known they sometimes might generate mistakes when solving some tasks. In this scenario, it is paramount to guarantee the correctness of the generated code to avoid any implementation

[4] https://developer.nvidia.com/tensorRT
[5] https://huggingface.co/jfaldanam/moostral-7B
[6] https://gitlab.com/jfaldanam-phd/moostral

Figure 6.2: Showcase of the graphical user interface for the proposed tool while generating a multi-objective optimization problem. The details of the implementation are not shown for formatting reasons; an example of the model output is available in Listing 6.1.

error. An example of these mistakes is what has been termed hallucinations [74], i.e., grammatically correct outputs stating incorrect facts.

Two validation approaches are considered to guarantee the correctness of the generated code. First, as done during the generation of synthetic problems, the LLM output is checked for syntactic errors by compiling the resulting code, and for basic functionality by running a set of unit tests that guarantee that the generated code will run within the framework.

Although not mandatory, the second validation method is recommended to verify the correctness of the generated code. To apply this method it is required that the domain expert provides a few values for the decision variables of the problem, as well as the expected output for these values. These validation inputs can either be manually calculated or, in the case of real-world problems, obtained from the results of previous physical experimentation. Figure 6.3 shows the pipeline to convert the user input into valid jMetal code.

By applying these validation steps, the correctness of the output cannot be guaranteed but, if it fails, a notification can be shown to the user confirming whether the generated implementation computes the same output for the provided variable values, thus giving it confidence in the implementation. This validation procedure is automatically integrated into the provided graphical user interface.

6.4 Evaluation of the Fine-Tuned Model

The quality of the proposed model is assessed with two types of experiments. In both cases, data not used for fine-tuning the model is used for the evaluation. First, the mathematical formulation of ten real-world problems is used as inputs and the generated code is analyzed. This experiment mimics the inputs and outputs during the fine-tuning phase. Second, the model is evaluated on natural language-based descriptions of problems instead of their mathematical formulation. Although the fine-tuned model has not been trained with this kind of input, it is based on a foundational model and, therefore, still retains some capabilities for natural language understanding. The goal is to analyze whether these capabilities can be leveraged to generate proper jMetal code.

6.4.1 Evaluation on Real-World Problems

Here, the quality of the implementation generated by the model is validated over a set of ten real-world problems. The selected problems are part of the real-world

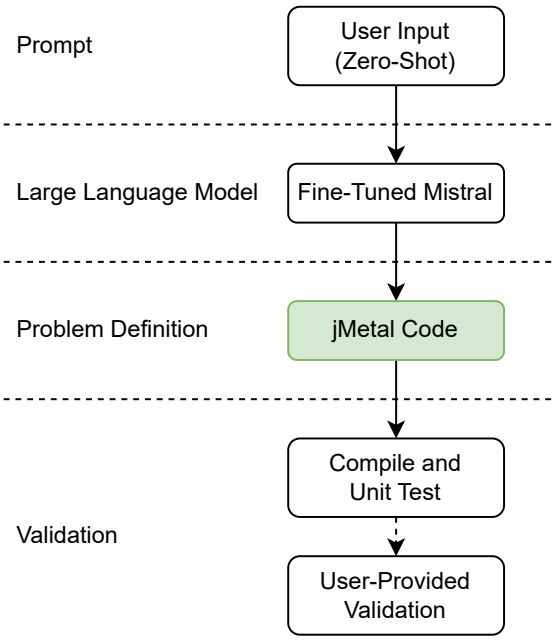


Figure 6.3: Pipeline for generating a problem implementation out of a user prompt and an optional validation step.

application (RWA) family of problems included in [167] and, which are detailed in Table 6.2.

First, a textual representation of the formulation of each of the problems is manually created and the fine-tuned model is used to generate an implementation for each of them. As their original jMetal implementation is also available, this is used to generate a set of three validation points for each application. Then, those points are used to validate the generated code through both of the methods described in the previous section.

From the ten problems available in the test suite, the approach described in this chapter is able to generate a class extending the jMetal $AbstractDoubleProblem$ class, which implements the $DoubleProblem$ interface, with the correct implementation in nine cases. The only case where the fine-tuned model has not been able to produce a valid implementation is for the packed bed latent heat thermal storage problem. The main characteristic of this problem is that its description (2227 symbols according to the Mistral tokenizer) is twice as long as the rest of the evaluated problems. When producing the code for this case, the language model did not place the commas at the end of each of the objectives, failing to

Problem	Discipline	Source
Design of honeycomb heat sinks	Structure engineering	[146]
Vehicle crashworthiness design	Structure engineering	[92]
Development of liquid-rocket single element injector	Structure engineering	[56, 153]
Production of synthesis gas	Chemical engineering	[52]
Performance of packed bed latent heat thermal storage	Materials engineering	[53]
Ultrahigh-strength steel milling parameters	Materials engineering	[163]
Development of water and oil repellent fabric	Materials engineering	[1]
Machining of wire electrical discharge	Electronics engineering	[119]
Design of ultra-wideband antenna	Electronics engineering	[26]

Table 6.2: Categorization and sources for the problems defined in the real-world applications test-suite [167].

compile it. This is due to the attention mechanism used by LMs[155] failing to remember that is writing Java code after generating a long formula. Indeed, one of the situations where LLMs often start making errors is when processing a very large number of symbols [100] (either as part of the input prompt or the generated ones), as it is common for the model to start misplacing them. A possibility to overcome this issue would be to have a wider representation of problems requiring larger descriptions in the training dataset.

6.4.2 Problem Implementation From Natural Language Descriptions

For this experiment, a paragraph defining a multi-objective problem, depicted in Listing 6.1, is provided as input to the model. The number of objectives is not explicitly indicated, but the paragraph includes a description of all of them in different ways. For the first and third objectives, the mathematical formulas are provided. The second objective formula, is however not explicitly stated; instead, it is described in the human text as:

User input:

Implement a multi-objective optimization problem where the first objective is the negative square root of the first variable, a second objective that is the sum of the sine of each of the variables minus six and a half, and a third objective that is the first variable minus the third one. The range of feasible values for each variable is between $-\pi$ and π.

Generated implementation:

```java
package xyz.jfaldanam;

import java.util.List;
import org.uma.jmetal.problem.doubleproblem.impl.AbstractDoubleProblem;
import org.uma.jmetal.solution.doublesolution.DoubleSolution;

public class ProblemY extends AbstractDoubleProblem {

  /** Constructor */
  public ProblemY() {
    int numberOfVariables = 3;
    numberOfObjectives(3);
    numberOfConstraints(0);
    name("ProblemY");

    List<Double> lowerLimit = List.of(-Math.PI, -Math.PI, -Math.PI);
    List<Double> upperLimit = List.of(Math.PI, Math.PI, Math.PI);

    variableBounds(lowerLimit, upperLimit);
  }

  /** Evaluate() method */
  @Override
  public DoubleSolution evaluate(DoubleSolution solution) {
    double x1 = solution.variables().get(0);
    double x2 = solution.variables().get(1);
    double x3 = solution.variables().get(2);

    solution.objectives()[0] = -Math.sqrt(x1);
    solution.objectives()[1] = (Math.sin(x1) + Math.sin(x2) +
        Math.sin(x3)) - 6.5;
    solution.objectives()[2] = x1 - x3;

    return solution;
  }
}
```

Listing 6.1: Generated jMetal code from the natural language description.

> "...a second objective that is the sum of the sine of each variable minus six and a half..."

The reason for doing so is to evaluate the natural language understanding of the model. The number of decision variables is intentionally not indicated, although their ranges of values are explicitly stated and there are mentions to the first and third variables.

Listing 6.1 contains the implementation generated by the model and it was validated to be correct. The model provided the right code for the explicit functions in the first and third objectives. Focusing on the variables, the model correctly understood that there are a total of three decision variables, even if the second one is not explicitly mentioned. With this understanding, it correctly defined the second objective and the ranges for the decision variables. This example showcases the generalization capabilities of LLMs, but more experimentation and representation of this kind of input representation in the fine-tuning dataset is required to guarantee results in this format.

Chapter 7

Algorithmic Recommendations Based on Semantic Knowledge

In this chapter, a recommendation tool for multi-objective metaheuristics, *recommoonder*, is presented. *recommoonder* is powered by a knowledge graph and provides a variety of interfaces to obtain different kinds of outputs, ranging from algorithmic recommendation to visualization methods or a generic entry point to query the knowledge graph. This tool provides a central tool that integrates all research and tools presented in this PhD thesis. *recommoonder* integrates all research and software tools developed during this PhD thesis, evaluating its main hypothesis.

7.1 Introduction

As mentioned in Chapter 1, end-users across various disciplines such as biologists, engineers and economists often encounter challenges when attempting to optimize multi-objective problems due to their limited expertise in metaheuristics. Consequently, they frequently resort to using the default settings of popular algorithms without customizing parameters to suit their specific real-world scenarios. As a result, a current and active research area involves developing tools that enable these users to efficiently identify suitable algorithms and configurations for addressing their unique problems.

Addressing this need, current research efforts are actively exploring methods for automating the tuning of metaheuristics parameters [73, 122] and even for the automatic design of algorithms [13]. These endeavors aim to streamline the process of parameter adaptation in metaheuristics, thereby enhancing their performance in tackling specific problem domains. However, it is important to note that these approaches often entail significant computational overhead. Nevertheless, researchers in this field are continuously advancing, striving to narrow the disparity between the intricacies of real-world problems and the efficacy of optimization techniques.

This chapter aims to validate the initial hypothesis of this PhD thesis: "Given previous knowledge on the relationship between a specific algorithmic configuration and the quality of the result of said algorithm solving a problem and given a similitude metric between two problems, it is possible to provide recommendations to non-expert users to choose an algorithmic configuration to efficiently solve a specific problem". To validate the hypothesis, this chapter aims to design and develop a recommendation tool for algorithms for solving multi-objective optimization problems by applying a knowledge-based approach based on semantic web technologies.

The primary contribution of this chapter lies in the development of a recommendation tool named *recommoonder*. This tool enables users to receive algorithmic recommendations for specific problems, drawing upon previous knowledge modeled after *moody* as outlined in Chapter 3. *recommoonder* undergoes validation through two different tests. The first one tests the recommender performance when looking for a problem that is available on underlying knowledge graph, while the second evaluates the performance on unknown problems.

Following this introduction, the chapter is organized as follows. Firstly, Section 7.2 provides an review of the state of the art in algorithmic recommendation. Section 7.3 presents an overview of the software architecture of the

recommender system and the provided interfaces. Finally, Section 7.4 validates *recommoonder* by evaluating the recommendations over benchmark and real-world problems and Section 7.5 discusses the results.

7.2 Literature review

[178] proposes a similar idea to the one described in this thesis proposing a case of study in flow shop scheduling, but focusing on mono-objective optimization and only algorithmic recommendation, without entering on the configuration of said algorithm.

Tian et al. [149] presents a recommendation system for metaheuristics based on deep learning, focusing on selecting the best kind of metaheuristic (genetic algorithm, ant colony optimization, etc.) based on features extracted from the mathematical decomposition of each objective into operators such as addition, subtraction, square, absolute value, etc. This representation of a function is then converted into reverse polish notation and used to train a deep recurrent neural network that predicts the best algorithm.

[93] introduces a landscape-aware automatic algorithm selection, which using a set of three multi-objective search algorithms NSGA-II [35], IBEA [176] and MOEAD/D [169] for a large-size of instances on a family of multimodal pseudo-boolean optimization problems known as $\rho mnk - landscape$ [156]. In a complementary study, [165] conducted an analysis on the same dataset which further visualizes the relationship between landscape features and algorithm performance.

[19] proposes a methodology to generate problem instances that behave vastly different for more than two algorithms simultaneously, which supports that finding the best configuration for a specific problem instance, as proposed in this work, is highly relevant.

In [122], the authors discuss on the conclusions the importance of tuning a metaheuristic to a specific problem and demonstrate how a tuned version of MOEA/D for a single problem performs better than a generic tuned version. The authors discuss as a future research line is researching what is the important feature that makes two problems similar.

Transfer learning is the application of knowledge obtained on one domain onto a different one [125, 120]. [50, 51] demonstrates the use of transfer learning to transfer algorithm configuration obtained for some problems and instances into new scenarios. This concept is used for the recommender presented in this

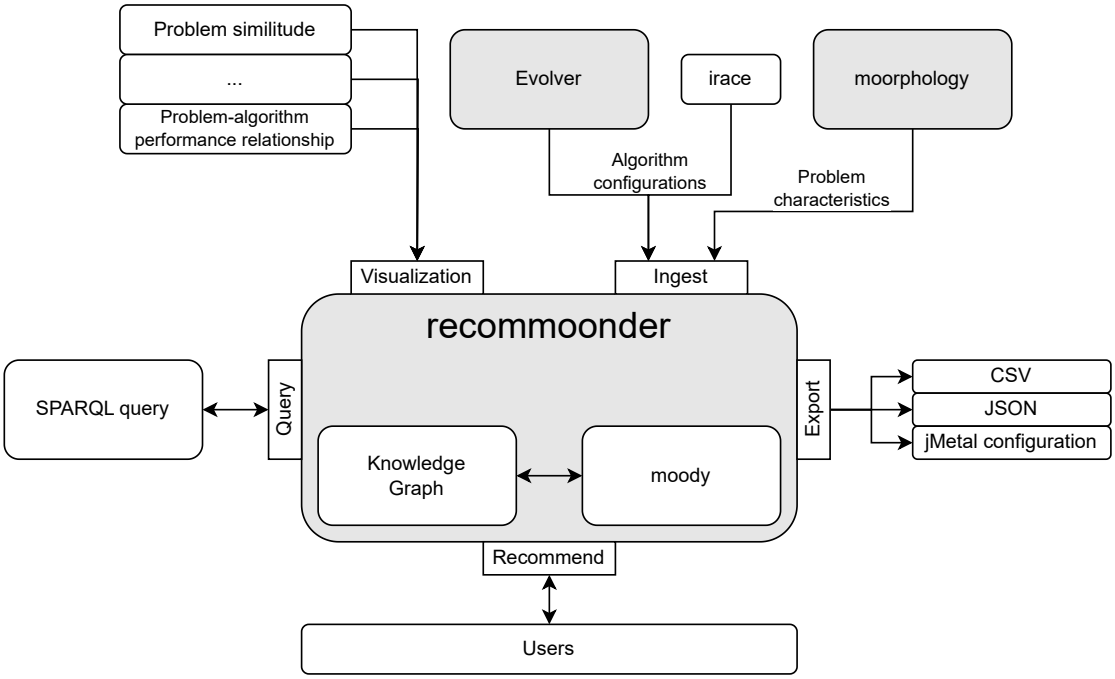


Figure 7.1: Software architecture of *recommoonder* with the main public interfaces.

thesis, which uses previous knowledge about the problems and configurations to transfer the best suitable one to a new domain, in this case a new multi-objective problem.

[108] discusses one possible issue that is faced by knowledge-based recommendation system. Known as the "cold-start" issue, it refers to the case when there is no previous information to base the recommendation on, for example in case of a problem where it's landscape is completely separated from the previously seen. In this cases, the fall back solution is to provide a default configuration for said algorithm.

7.3 Architecture

The proposed tool, *recommoonder*, is a recommendation engine, powered by a knowledge graph, that provides a set of interfaces to allow the ingestion, visualization and export of data, a query interface for the knowledge graph and, finally, the recommendation engine itself. Figure 7.1 shows this architecture.

Each of these interfaces will be explored in more detail in the following sections.

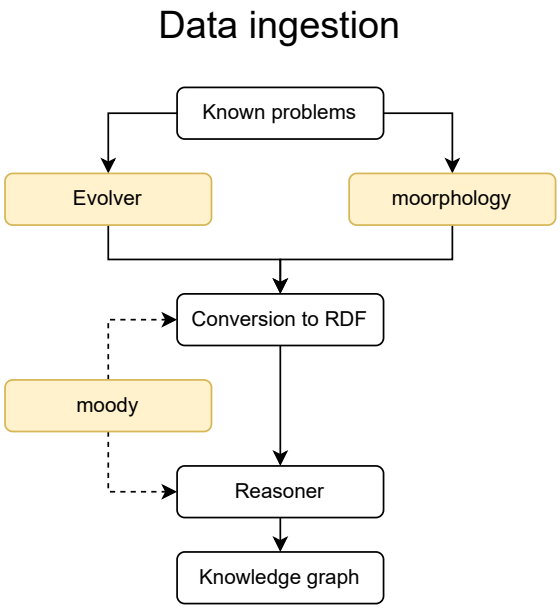


Figure 7.2: Ingestion pipeline for *recommoonder*.

7.3.1 Data ingestion

This interface provides mapping functions to convert data from different standard formats into RDF, following the semantic model provided by *moody* and described in Chapter 3. *recommoonder* requires two types of data: an analysis of the landscape of multi-objective optimization problems and a set of algorithmic configurations, that solve those problems based on a series of quality indicators.

For data on landscape analysis, a connector to *moorphology*, described in Chapter 4, is provided. Through this interface new problems can be added to the knowledge graph directly from the output of *moorphology*.

Chapter 5 provided in this thesis promotes the use of automatic configuration of metaheuristic to generate the knowledge that will be used for recommendation. There are connectors available for *Evolver*, the tool presented in this thesis, and irace [103], representing an existing mature and popular tool for auto-configuration [37], used in works such as [11, 12, 115].

Once data is converted to RDF, a reasoner is executed over it using the SWRL rules defined in Section 3.4 to guarantee that the algorithmic configurations are valid before inserting them into the knowledge graph.

An overview of the ingestion process is available in Figure 7.2.

7.3.2 Recommendation engine

The recommendation engine is built as a knowledge graph that integrates previous knowledge, as mentioned in the previous section, powered by semantic web technologies. As such, a recommendation is, in essence, the response to a specific query to the knowledge base that retrieves the required knowledge for the user to be more informed to make a decision. The knowledge graph is stored as a RDF graph which can be queried using the SPARQL query language.

The recommendation process is outlined in Figure 7.3. To explain this process, let us consider a scenario where a user aims to tackle a continuous multi-objective problem without prior knowledge of its characteristics or suitable algorithms to solve it.

Initially, the user employs *moorphology* to derive a set of landscape characteristics that describe the problem. With this characterization, the user then consults *recommoonder* to identify the optimal configuration for addressing problems with similar characteristics. Upon finding a suitable configuration, the user can export that configuration as a jMetal execution command that will solve the target problem efficiently.

In cases where no suitable configuration is identified (or if it falls below a confidence threshold), the user may resort to a more computationally intensive approach, such as utilizing *Evolver* to auto-configure an algorithm to solve his target problem. Afterwards, the user can submit its results to *recommoonder* to enhance the knowledge graph and improve future recommendations.

7.3.3 Ancillary interfaces

In addition to the main functionalities, *recommoonder* provides ancillary interfaces to facilitate different kinds of analysis on the knowledge graph.

Firstly, a SPARQL endpoint is provided, offering direct access to the knowledge graph, which allows users to explore the stored graph by employing the SPARQL query language, enabling them to extract specific information tailored to their needs. This endpoint allows the user to input any valid SPARQL query. However, users can also find some sample SPARQL queries in Appendix B, which can be used as-is or modified according to their needs

Secondly, a visualization engine is incorporated, equipped with common graphs for data within the knowledge graph. These visualizations include but are not limited to: the similarity of problems or the correlation between algorithmic configurations and the performance in a specific problems. These visual aids en-

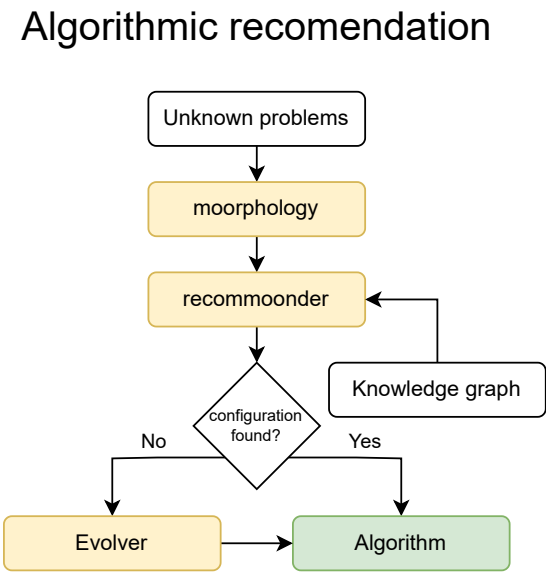


Figure 7.3: Recommendation pipeline for *recommoonder*.

hance the understanding of the data, making it more accessible and interpretable for users analyzing it.

Finally, *recommoonder* offers an interface for exporting the knowledge graph, or part of it, in various standard formats such as comma-separated values (CSV) or JavaScript Object Notation (JSON). While exporting in these formats may result in the loss of some information present in the RDF representation of the knowledge graph, it proves invaluable for users and researchers seeking to apply artificial intelligence or other analytical techniques to the data. Additionally, the system provides an export option tailored specifically for jMetal, providing users with the precise commands required to execute a specific configuration of an algorithm, facilitating the application of knowledge derived from *recommoonder* in real-world scenarios.

7.4 Evaluation

To assess the effectiveness of the recommendations provided by the recommender, two distinct scenarios are proposed for evaluation.

Problem	Distance
DTLZ1_2D	2.175
DTLZ1_3D	5.573
DTLZ3_2D	6.115
ZDT6	6.487
DTLZ5_2D	6.654

Table 7.1: Top 5 Problems with more similarity to the anonymized problem (DTLZ1_2D.

7.4.1 Evaluation on known problems

Firstly, the recommender's capability to identify optimal configurations for known problems is examined. In this scenario, a problem is selected from the knowledge graph, and its characteristics are anonymized through the computation of a new sampling, so that there is no exact match with the stored sampling. Subsequently, the recommender is tasked with suggesting the best configuration to address this anonymized problem. The expectation is that the recommender will identify the most effective solution known for the chosen problem. In Figure 7.4, the distances between all problems are depicted. Most problems exhibit a few close neighbors, while the majority of its neighbors display similarity distances closer to the average.

For this evaluation, the DTLZ1 problem with two objectives has been selected, and a new random sampling is generated and characterized through by using *moorphology*. With this characterization and using as reference the hypervolume quality indicator, the recommendation system is tasked on finding the best solution for a problem with similar characteristics.

In Table 7.1, the anonymized problem is most similar with itself by a significant difference, showcasing high confidence on the configurations that will be provided by the recommendation system, followed by the 3 objective variant of the same problem. Following a greedy approach, the best configuration to solve the most similar problem is selected. Figure 7.5 showcases the front obtained from both the default NSGA-II configuration and the recommended configuration on 10000 evaluations.

7.4.2 Evaluation on unknown problems

In the next step, the recommender's performance is evaluated when presented with some real-world problems unfamiliar to the recommendation system. The recommendation system is provided with a sampling of an unknown problem

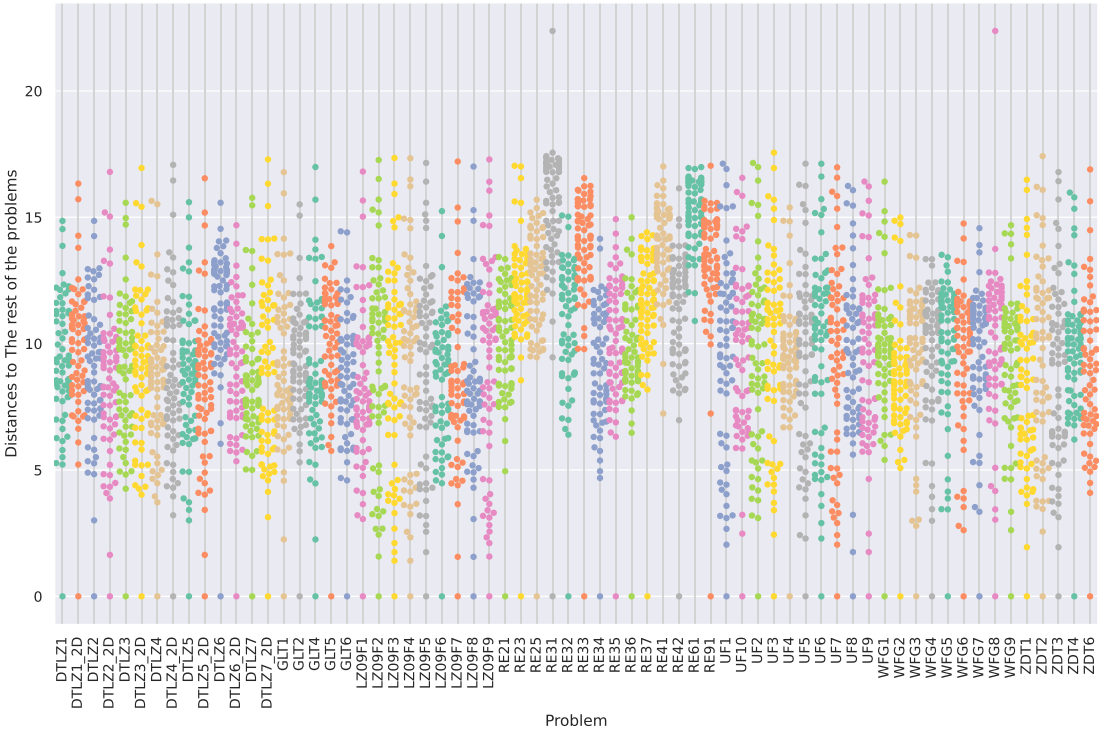


Figure 7.4: Swarmplot of the distance to each problem to the rest of known problems.

and seek a solution through identifying similarities with other known problems. By comparing the obtained configuration with a default NSGA-II configuration, the effectiveness of the recommendation is assessed for solving the unfamiliar problem. It's essential to acknowledge the possibility of not finding an optimal configuration due to insufficient information in the knowledge graph, commonly referred to as the "cold-start issue" [108].

For this evaluation, three examples of the real-world application (RWA) family [167] will be used. The RWA problems are chosen because of their real-world nature, and their family of problems is not part of the knowledge graph. The selected problems for this evaluation are the design a single element injector of liquid-rocket engine problem (Goel2007) [56], the crashworthiness design of vehicles problem (Liao2008) [92] and the optimization of milling parameters for ultrahigh-strength steel problem (Xu2020) [163].

After sampling the Goel2007 problem, the most similar problems can be seen in Table 7.2, being RE37 [147, 153] the closest. Upon further inspection into the RE37 problem, it is actually a different formulation of the design of the injector of a liquid-rocket engine. This variant is also included on the RWA family under

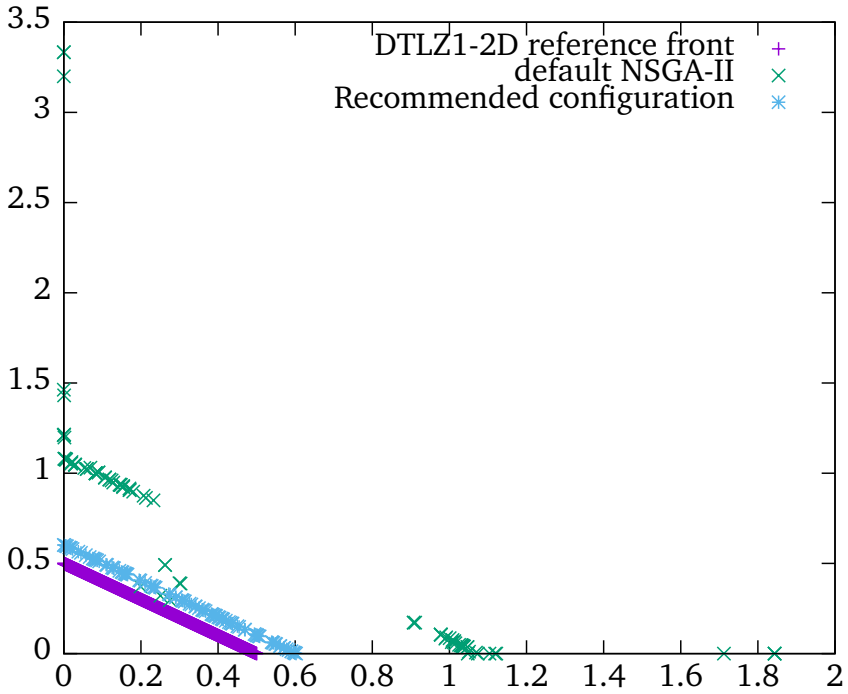


Figure 7.5: Evaluation of the recommended configuration versus the default NSGA-II configuration for solving the DTLZ1 problem in 10000 evaluations.

the name of Vaidyanathan2004. A closer inspection on the characteristics of both problems showcases their similarity mainly in the variable space as the difference between them is in the number of objectives.

On Figure 7.6, the populations of the default NSGA-II and the recommended algorithm after 10000 evaluations for solving Goel2007. The recommended configuration has converged more in the limited evaluation budget and achieves better diversity.

Problem	**Distance**
RE37	2.177
WFG4	8.656
RE33	8.862
GLT5	8.880
RE21	9.045

(a) Goel2007

Problem	**Distance**
RE34	2.954
RE21	5.965
GLT6	5.993
LZ09F6	6.371
DTLZ6_2D	6.469

(b) Liao2008

Problem	**Distance**
RE35	9.7141
RE37	9.811
WFG4	10.012
RE21	10.469
GLT5	10.666

(c) Xu2020

Table 7.2: Top 5 Problems with more similarity to each real-world problem.

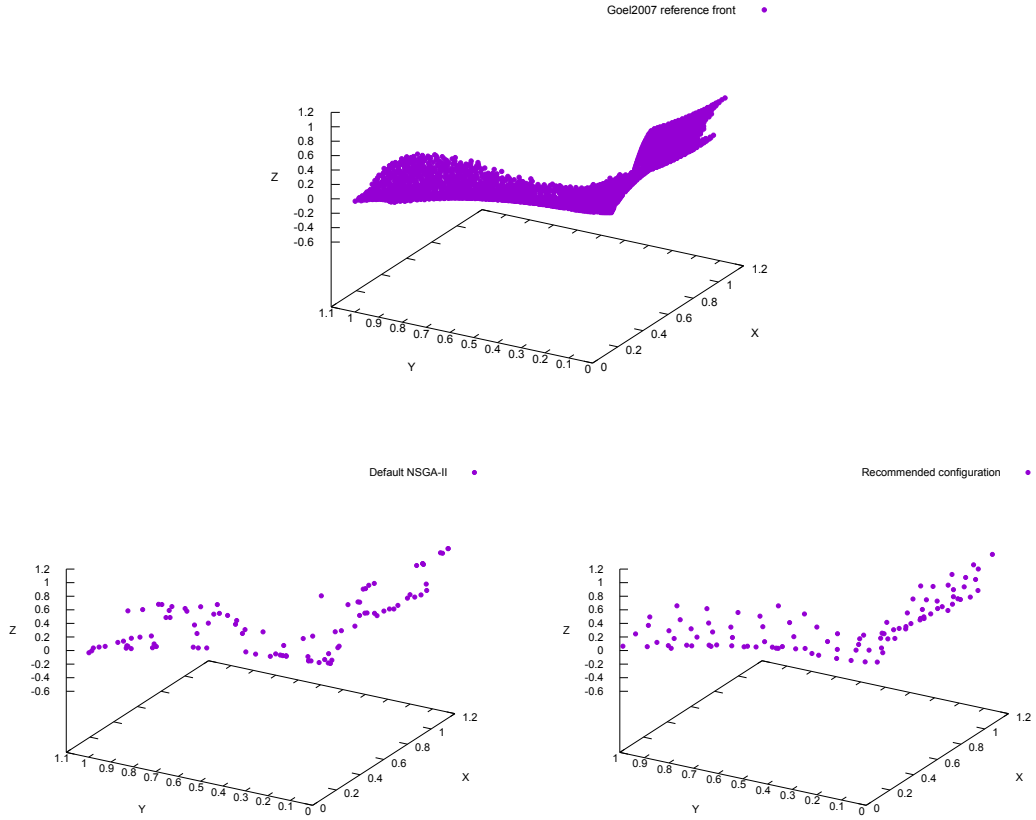


Figure 7.6: Reference front of Goel2007 (top), front obtained by NSGA-II with standard setting (bottom left), and front obtained by *recommoonder* (bottom right).

Focusing on Liao2008, the closest problem found after the sampling is the RE34, as seen in Table 7.2. Reviewing the sources of both problems [167, 147, 47], they are actually the same problem, solving the crashworthiness design of vehicles [92].

In the case of Liao2008, there is a big difference between the Pareto fronts obtained by the default and recommended configurations on a budget of 10000 evaluations, as shown in Figure 7.7. As commented previously, Liao2008 was actually a problem already available inside the knowledge graph as RE34. As the knowledge graph was populated using a auto-configuration approach, it is expected of this configuration to have a high performance on their respective problems.

After analyzing the most similar problems to Xu2020, there is no single close

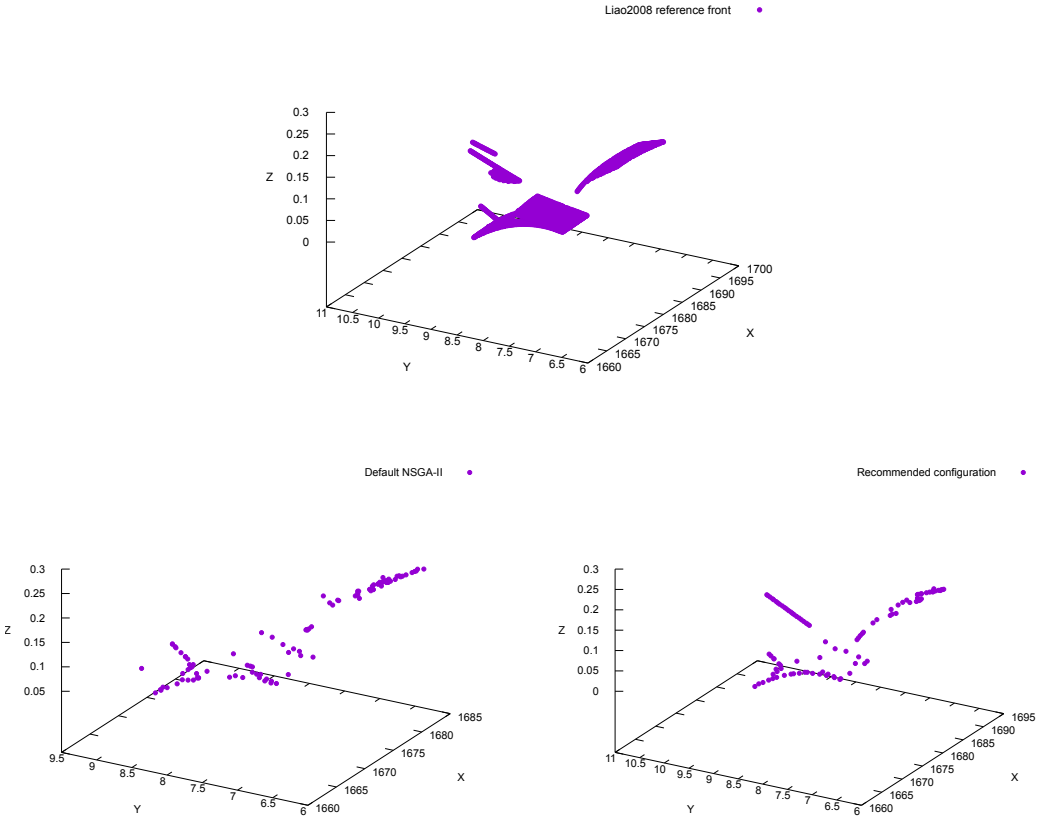


Figure 7.7: Reference front of Liao2008 (top), front obtained by NSGA-II with standard setting (bottom left), and front obtained by *recommoonder* (bottom right).

problem that is significantly more similar than the rest. This lack of a relationship with the studied problems means that is possible that a good configuration can not be found for Xu2020. Following a greedy approach, the RE35 is selected as it is the closest problem, as shown in Table 7.2. While Xu2020 optimizes the milling parameters for ultrahigh-strength steel, the RE35 [147, 47] is also a real-world problem that optimizes the design of a speed reducer, a simple gear-box that can be used in a light airplanes.

Figure 7.8 shows the Pareto fronts obtained by the default NSGA-II configuration and the recommendation provided by *recommoonder*. In this case there is a less noticeable difference between both fronts, probably cause by the low similarity to the closest problem.

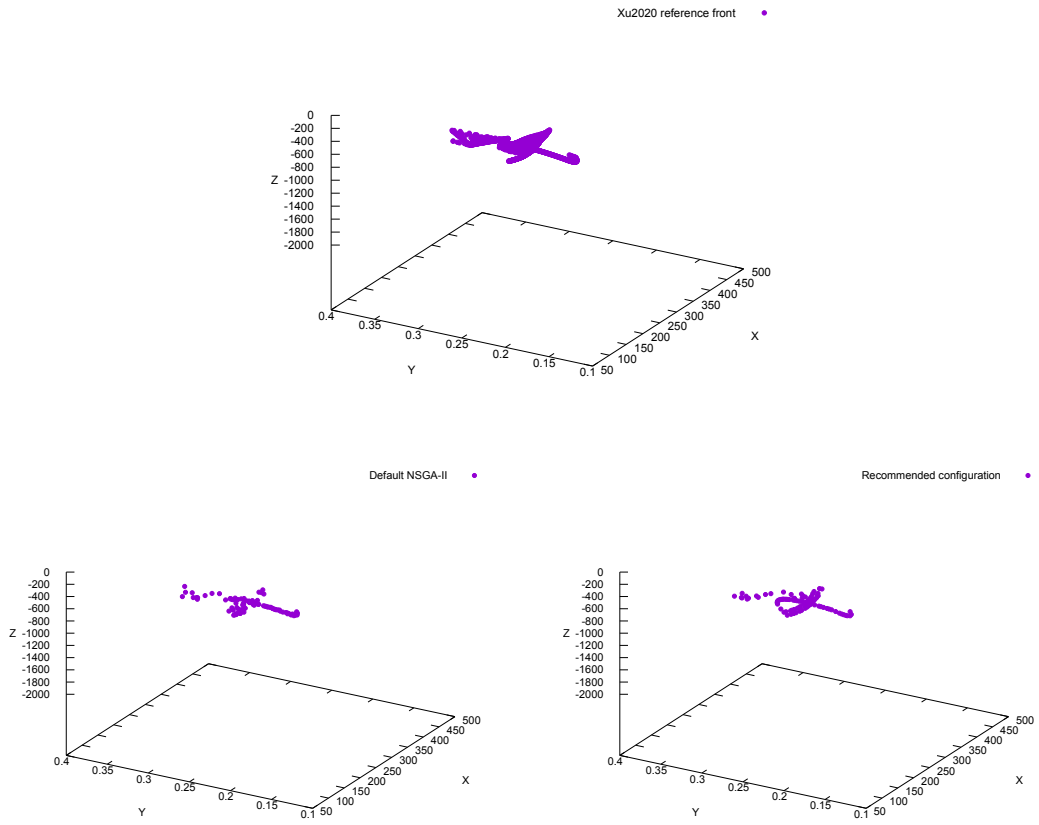


Figure 7.8: Reference front of Xu2020 (top), front obtained by NSGA-II with standard setting (bottom left), and front obtained by *recommoonder* (bottom right).

7.5 Discussion

After this experimentation, a lot of overlap has been found between the existing families of real-world problems presented as benchmarks for multi-objective optimization. Three of the ten problems of the RWA family proposed by Zapotecas-Martínez et al. (2023) [167] are also found in a previous study by Tanabe and Ishibuchi (2020) [147]. The RWA paper claims:

> "In this regard, some investigators have summed up some real-world applications coming from different research lines. (Tanabe and Ishibuchi, 2020) reported a set of real-world problems taken from different disciplines to evaluate the performance of multi-objective algorithms. (...) This work presents a completely distinct set of multi-objective optimization problems with real-world constraints and some bench-

mark results."

However, this chapter found empirically, and later confirmed by reviewing the sources of each problem, that Liao2008 and Vaidyanathan2004 are the same problem previously presented as RE34 and RE37, respectively. Another of the problems, Goel2007, is also a variant of RE37.

Additionally, there seems that real-world problems are often more similar with other real-world problems that to benchmark problems, as seen in Table 7.2. Benchmark problems are often designed to be easily configurable, scalable on the number of variables, or objectives, or extremely hard. While those are useful features for benchmarking algorithms, this kind of problems are not similar to real-world problems.

In real world scenarios, there's often a recurring necessity to solve numerous instances of the same optimization problem. For instance, consider a engineering company addressing the optimization of the milling parameters for ultrahigh-strength steel, while testing with different steel variants that modify some of the parameters of the objective functions.

The same approach presented on this chapter could be used by creating a custom knowledge graph containing the solved instances of the optimization of the milling parameters problem that the company are solving daily. Once they have generated a knowledge base, the conventional generic similarity measure between problems can be replaced with a domain-specific distance metric, which better captures similarities among problem instances. This enables the development of custom recommender system to meet specific requirements on domain specific optimization problems.

Chapter 8

Conclusions and Future Work

This chapter presents the conclusions and prospective future work derived from this dissertation. It summarizes the main findings and highlights their relevance in each of the related fields. Based on the findings and limitations identified, the focus then shifts to potential areas for future research.

8.1 Conclusions

To conclude this PhD thesis, some conclusions are provided on each of the areas that have been discussed.

8.1.1 Semantic modeling in the multi-objective optimization domain

This PhD thesis proposes an ontology-based framework for standardization in multi-objective optimization, focusing on evolutionary algorithms. *moody*, the main ontology guiding the framework, covers aspects from the formalization of evolutionary algorithms and their parameters and multi-objective problems with their landscape characteristics.

moody is developed in OWL 2 and is linked to external ontologies like BIGOWL, OPTION and DMOP. A reference implementation to ingest configurations of evolutionary algorithms is provided, converting the output of the auto-configuration tool irace to the standard format RDF.

Four use cases have been provided to validate the framework. These use cases are the enhancement of auto-configuration tools via the *moody* framework, data integration from auto-configuration tools or other sources, querying of the knowledge graph to obtaining valuable insights and exporting this knowledge into optimization frameworks used in real world applications, like pagmo [14] from the European Space Agency.

To provide landscape characteristics of multi-objective problems to the proposed knowledge graph, an implementation of these metrics is provided as the software project *moorphology*. To validate the selected set of metrics, an evaluation of the stability of the characteristics over the COCO bi-objective suite is provided, accompanied of an implementation of the suite in jMetal. Additionally, a study to evaluate the quality of the set of features to characterize multi-objective optimization problems is provided.

8.1.2 Auto-configuration of metaheuristics

A study is presented in which the NSGA-II algorithm is used as a meta-optimizer, i.e., as a tool that, given a set of problems as training set, is aimed at finding the best configurations from a set of NSGA-II parameters and components. Utilizing a simple encoding scheme and leveraging features existing in jMetal, the proposal is entirely developed within jMetal, thus eliminating the need for any external tools.

Several experiments have been defined to validate this proposal considering two scenarios and three experiments to cover both automatic search of NSGA-II designs for single and multi-problem training sets. The outcomes of these experiments reveal that the meta-optimizer is able of finding configurations of NSGA-II that successfully achieve the defined goals.

After validating the approach, *Evolver* is presented as a software package aimed at the meta-optimization of multi-objective metaheuristics. By defining the automatic search of configurations for multi-objective optimizers as a multi-objective problem, *Evolver* facilitates the user to find algorithm variants that are tailored to a number of optimization problems used as training set. This tool offers a number of representative multi-objective metaheuristics which are highly configurable (NSGA-II, MOEA/D, SMS-EMOA, MOPSO).

Evolver is implemented in Java and it is based on the jMetal framework, so a large amount of existing metaheuristics can be used as meta-optimizers. Users familiarized with jMetal will be comfortable with *Evolver* and have the opportunity to use it a tool for research in the line of automatic metaheuristic tuning. The provided GUI allows non-expert users to easily set and run a meta-optimization execution.

How *Evolver* works is illustrated by considering an example representing a typical scenario in which an engineer intends to find a variant of NSGA-II to solve a given kind of problems.

Additionally, a novel alternative approach is proposed by applying Quality-Diversity for automatic configuration of metaheuristics. The application of this optimization techniques provides a set of diverse algorithmic configuration that are ensembled to improve the robustness and generalization of the individual configurations and is evaluated over a large set of benchmark problems.

8.1.3 Automatic implementation of multi-objective optimization problems

The challenge of bridging the natural language or textual representation of the formulation of multi-objective optimization problems into executable code is addressed, designing a tool that automatizes the implementation process.

The key contributions lies in the fine-tuning of an LLM over a synthetic dataset of multi-objective problems generated by a custom problem generator provided alongside the model. The proposed model effectively translates the textual representation of the formulation of a optimization problem into their

equivalent implementation in the jMetal framework, and the model is empirically validated using a suite of ten real-world multi-objective problems.

Additionally, the trained model is embedded within a tool with a graphical user interface and a set of validation steps to guarantee the correctness of the provided implementation without requiring the high-end computational power often associated with LLMs. Both the model weights and the tool are released under an open license.

8.1.4 Automatic recommendation of multi-objective optimization algorithms

The final contribution is the development of a tool named *recommoonder*. This tool integrates the capabilities of the previous software packages into a recommendation engine in a user-friendly package. It is designed to assist non-expert users in selecting algorithm configurations that surpass standard default settings.

The tool is then evaluated on both known and unknown problems and confirms empirically how the similarity measurements that are found in Chapter 4 are relevant to the recommendation of specific algorithmic configurations.

8.2 Future Work

This sections enumerates potential lines of work to continue the research presented in this PhD thesis. The source code for all the tools presented in this PhD thesis is open source under a permissive license for anyone interested in continuing any of the presented research lines.

- The *moody* ontology can be extended by including other metaheuristics, such as particle swarm optimization, and new problems, including discrete optimization problems. In this sense, working with real world problems is particularly interesting, but proposes an special challenge as usually the Pareto front is unknown.
- An in-depth analysis on whether the set of landscape characteristics presented in this thesis is the optimal set of landscape characteristics, how they affect algorithmic performance and a quantitative analysis on how they relate to problem similitude are open research lines.
- The proposed metric for calculating the similarity between multi-objective optimization problems, which currently relies on measuring the distance

between their landscape characteristics, can be enhanced through the integration of machine learning techniques. These techniques could be employed to dynamically assign varying weights to each characteristic, depending, for example, of the algorithm used.

- Another interesting line involves a deeper study into the similarity between the problems of a single family, what characteristics are affecting that similarity and the biases introduced by the authors.
- On the topic of auto-configuration, a study that examines the extent to which the number of evaluations of the meta-optimizer can be reduced in the search while the resulting NSGA-II designs are still able solve the problems efficiently. A related study is finding to which extent the computational budget for the configurations being evaluated can be reduced while still obtaining good configurations. This latter option is specially interesting on computationally expensive problems.
- Adapt *Evolver* to support optimization on problems with an unknown Pareto front. This involves the use of quality metrics for the meta-optimization approach that doesn't required a reference front, such as the hypervolume, and dynamically adjusting the reference front during the search process. This second option involves using a external archive of solutions that will be re-evaluated when updating the reference.
- On the automatic implementation of multi-objective optimization problems, the training of LLMs to generate output for other optimization frameworks like PlatEMO or Pymoo is an open research line. This extension in PlatEMO would open up the tool to a greater number of users. Other research lines to improve the efficiency of the model are the quantization of the model or the pre-training with custom tokenizers designed to recognize framework specific symbols. This latter would allow reducing the number of tokens required to generate valid code, improving both latency and inference cost.
- A different approach to improving the proposed LLM is the study on the use of semantic technologies to inject domain knowledge into the model, intending to generate problem implementations from informal natural language descriptions or closer to the language used by humans.
- Continuing the research on the use of quality-diversity optimization with a study on the application of diverse ensemble on real-world problems is also an open research line.

Appendix A

Design Spaces of Metaheuristics

This appendix offers a comprehensive description of the design spaces of the configurable multi-objective metaheuristics implemented in *Evolver*, namely the NSGA-II (Non-dominated Sorting Genetic Algorithm II), the SMS-EMOA (S-Metric Selection Evolutionary Multi-Objective Algorithm), the MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition) and the MOPSO (Multi-Objective Particle Swarm Optimization) algorithm. Table A.1 includes the component and parameters of the NSGA-II algorithm. The parameters and components of the SMS-EMOA algorithm are available in Table A.2. Table A.3 shows the components and parameters available for the configurable implementation of MOEA/D. A highlight of this implementation is the variation parameter, allowing the meta-optimizer to find configurations of both the original MOEA/D [169] and the implementation with differential evolution (MOEA/D-DE) [89]. Table A.4 shows the components and parameters for a configurable implementation of MOPSO, representing a different set of metaheuristics.

Parameter/Component	Type	Domain	Dependency
algorithmResult	c	{externalArchive, population}	
populationSizeWithArchive	i	[10, 200]	algorithmResult == externalArchive
externalArchive	c	{crowdingDistanceArchive, unbounded}	algorithmResult == externalArchive
offspringPopulationSize	i	[1, 400]	
selection	c	{tournament, random}	
selectionTournamentSize	i	[2, 10]	selection == tournament
createInitialSolutions	c	{random, latinHypercubeSampling, scatterSearch}	
crossover	c	{SBX, BLX_ALPHA, wholeArithmetic}	
crossoverProbability	r	[0.0, 1.0]	
crossoverRepairStrategy	c	{random, round, bounds}	
sbxDistributionIndex	r	[5.0, 400.0]	crossover == SBX
blxAlphaCrossoverAlphaValue	r	[0.0, 1.0]	crossover == BLX_ALPHA
mutation	c	{uniform, polynomial, linkedPolynomial, nonUniform}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	mutation $\in$ {polynomial, linkedPolynomial}
uniformMutationPerturbation	r	[0.0, 1.0]	mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	mutation == nonUniform

Table A.1: Design space of the configurable NSGA-II algorithm in *Evolver*.

Parameter/Component	Type	Domain	Dependency
algorithmResult	c	{externalArchive}	
populationSizeWithArchive	i	[10, 200]	algorithmResult == externalArchive
externalArchive	c	{crowdingDistanceArchive, unbounded}	algorithmResult == externalArchive
offspringPopulationSize	i	1	
selection	c	{tournament, random}	
selectionTournamentSize	i	[2, 10]	selection == tournament
createInitialSolutions	c	{random, latinHypercubeSampling, scatterSearch}	
crossover	c	{SBX, BLX_ALPHA, wholeArithmetic}	
crossoverProbability	r	[0.0, 1.0]	
crossoverRepairStrategy	c	{random, round, bounds}	
sbxDistributionIndex	r	[5.0, 400.0]	crossover == SBX
blxAlphaCrossoverAlphaValue	r	[0.0, 1.0]	crossover == BLX_ALPHA
mutation	c	{uniform, polynomial, linkedPolynomial, nonUniform}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	mutation $\in$ {polynomial, linkedPolynomial}
uniformMutationPerturbation	r	[0.0, 1.0]	mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	mutation == nonUniform

Table A.2: Design space of the configurable SMS-EMOA algorithm in *Evolver*.

Parameter/Component	Type	Domain	Dependencies
neighborhoodSize	i	[10, 200]	
maximumNumberOfReplacedSolutions	i	[1, 5]	
aggregationFunction	c	{tschebyscheff, weightedSum, PBI, modifiedTschebyscheff}	
normalizeObjectives	c	{TRUE, FALSE}	
epsilonParameterForNormalizing	r	{1.0e-8, 25.0}	normalizeObjectives == TRUE
pbiTheta	r	{1.0, 200.0}	aggregationFunction == PBI
algorithmResult	c	{population, externalArchive}	
externalArchive	c	{crowdingDistance, unbounded}	algorithmResult == externalArchive
createInitialSolutions	c	{random, latinHypercubeSampling, scatterSearch}	
selection	c	{populationAndNeighborhoodMatingPoolSelection}	
neighborhoodSelectionProbability	r	[0.0, 1.0]	
variation	c	{crossoverAndMutation, differentialEvolution}	
mutation	c	{uniform, polynomial, linkedPolynomial, nonUniform}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	mutation $\in$ {polynomial, linkedPolynomial}
uniformMutationPerturbation	r	[0.0, 1.0]	mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	mutation == nonUniform
crossover	c	{SBX, BLX_ALPHA, wholeArithmetic}	variation == crossoverAndMutation
crossoverProbability	r	[0.0, 1.0]	
crossoverRepairStrategy	c	{random, round, bounds}	
sbxDistributionIndex	r	[5.0, 400.0]	crossover == SBX
blxAlphaCrossoverAlphaValue	r	[0.0, 1.0]	crossover == BLX_ALPHA
CR	r	[0.0, 1.0]	variation == differentialEvolution
F	r	[0.0, 1.0]	variation == differentialEvolution

Table A.3: Design space of the configurable MOEA/D algorithm in *Evolver*. Types: (c)ategorical, (i)nteger, (r)eal. (PBI; penaltyBoundaryIntersection)

Parameter/Component	Type	Domain	Dependecies
swarmSize	i	[2, 200]	
leaderArchive	c	{crowdingDistanceArchive, hypervolumeArchive, SSDA}	
swarmInitialization	c	{random, latinHypercubeSampling, scatterSearch}	
velocityInitialization	c	{default, SPSO2007, SPSO2011}	
mutation	c	{uniform, polynomial, linkedPolynomial, nonUniform}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	mutation $\in$ {polynomial, linkedPolynomial}
uniformMutationPerturbation	r	[0.0, 1.0]	mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	mutation == nonUniform
frequencyOfApplicationOfMutationOperator	c	[1.0, 1.0]	
inertiaWeightComputingStrategy	c	{constantValue, random, LI, LD}	
weight	r	[0.1, 1.0]	inertiaWeightStrategy == constantValue
weightMin	r	[0.1, 0.5]	inertiaWeightStrategy $\in$ {random, LI, LD}
weightMax	r	[0.5, 1.0]	inertiaWeightStrategy $\in$ {random, LI, LD}
velocityUpdate	c	{defaultVelocityUpdate, constrainedVelocityUpdate, SVU}	
c1Min	r	[1.0, 2.0]	
c1Max	r	[2.0, 3.0]	
c2Min	r	[1.0, 2.0]	
c2Max	r	[2.0, 3.0]	
globalBestSelection	c	{tournament, random}	
selectionTournamentSize	i	[2, 10]	globalBestSelection == tournament
velocityChangeWhenLowerLimitIsReached	r	[-1.0, 1.0]	
velocityChangeWhenUpperLimitIsReached	r	[-1.0, 1.0]	

Table A.4: Design space of the configurable MOPSO algorithm in *Evolver*. Types: (c)ategorical, (i)nteger, (r)eal. (SSDA; spatialSpreadDeviationArchive, LI; linearIngreasing, LD; linearDecreasing, SVU; SPSO2011VelocityUpdate)

Appendix B

Sample SPARQL Queries

This appendix provides all the example queries for the *moody* framework, mainly described in Section 3.4. Listing B.1 includes the SPARQL query defined to obtain the parameter configuration from a specific experiment, with the results shown in Table B.1. Which algorithmic configuration is most effective in solving a specific problem, based on a particular quality indicator, can be retrieved by using Listing B.2 with example outputs in Table B.2. A derived query to check for the best performing algorithm for problems with similar characteristics can be seen in Listing B.3, with results on Table B.3, for problems with a disconnected front. Listing B.4 demonstrate how to query for configurations that are compatible with pagmo. From Section 4.4, Listing B.6 showcases the implementation of similarity metric (Listing B.5) between multi-objective problems based on their landscape characteristics.

```
PREFIX moody: <https://w3id.org/moody#>
SELECT DISTINCT ?parameter ?value
WHERE {
    VALUES ?experiment { moody:Experiment_NSGAII_DTLZ1_GECCO19 } .
    ?experiment moody:parameterValue ?parameter_value .
    ?parameter_value moody:valueOfParameter ?parameter .
    ?parameter_value ?property ?value .
    ?property rdfs:subPropertyOf moody:parameterValueProperty .
}
```

Listing B.1: SPARQL Query Q1, extraction the parameters of one specific experiment. **moody:Experiment_NSGAII_DTLZ1_GECCO19** is an example of the URI of an *Experiment*.

Result:

?parameter	?value
moody:Parameter_AlgorithmResult	externalArchive
moody:Parameter_BlxAlphaCrossoverAlphaValue	0.5906
moody:Parameter_CreateInitialSolutions	latinHypercubeSampling
moody:Parameter_CrossoverProbability	0.9874
moody:Parameter_CrossoverRepairStrategy	bounds
moody:Parameter_Crossover	BLX_ALPHA
moody:Parameter_ExternalArchive	crowdingDistanceArchive
moody:Parameter_MaximumNumberOfEvaluations	25000
moody:Parameter_MutationProbability	0.0015
moody:Parameter_MutationRepairStrategy	random
moody:Parameter_Mutation	polynomial
moody:Parameter_OffspringPopulationSize	200
moody:Parameter_PolynomialMutationDistributionIndex	158.05
moody:Parameter_PopulationSizeWithArchive	20
moody:Parameter_PopulationSize	100
moody:Parameter_ProblemName	dtlz.DTLZ1
moody:Parameter_ReferenceFrontFileName	DTLZ1.csv
moody:Parameter_SelectionTournamentSize	9
moody:Parameter_Selection	tournament
moody:Parameter_Variation	crossoverAndMutationVariation

Table B.1: Results for SPARQL Query Q1, extraction the parameters of one specific experiment.

```
PREFIX moody: <https://w3id.org/moody#>
SELECT DISTINCT
    ?experiment (AVG(?exp_value) as ?value)
WHERE {
    VALUES ?problem { moody:Problem_ZDT2 } .
    VALUES ?indicator {
        moody:QualityIndicator_HyperVolume
    } .
    ?problem_resolution rdf:type
        moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment
        ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?problem_resolution moody:indicatorValue
        ?indicatorValue .
    ?indicatorValue moody:valueOfIndicator
        ?indicator .
    ?indicatorValue moody:hyperVolumeValue
        ?exp_value .
}
GROUP BY (?experiment)
ORDER BY ASC(?value)
LIMIT 5
```

Listing B.2: SPARQL Query Q2, recover the experiments that better resolve a problem according to a specific quality indicator. **moody:Problem_ZDT2** is the URI of a *Problem* and **moody:QualityIndicator_HyperVolume** is the URI of a *Quality Indicator*.

Result:

?experiment	?value
moody:Experiment_2246	3.28790E-1
moody:Experiment_2235	3.28792E-1
moody:Experiment_2241	3.28801E-1
moody:Experiment_1935	3.28809E-1
moody:Experiment_2266	3.28814E-1

Table B.2: Results for SPARQL Query Q2, recover the experiments that better resolve a problem according to a specific quality indicator.

```
PREFIX moody: <https://w3id.org/moody#>
PREFIX bigowl: <http://www.khaos.uma.es/lod/bigowl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT
    ?experiment
    (SAMPLE(?problem) as ?problem)
    (MIN(?exp_value) as ?value)
WHERE {
    VALUES ?geometry { "disconnected"^^xsd:string  } .
    VALUES ?indicator { moody:QualityIndicator_HyperVolume } .
    ?problem_resolution rdf:type moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?problem rdf:type bigowl:Problem .
    ?probem moody:geometry ?geometry .
    ?problem_resolution moody:indicatorValue ?indicatorValue .
    ?indicatorValue moody:valueOfIndicator ?indicator .
    ?indicatorValue moody:hyperVolumeValue ?exp_value .
}
GROUP BY (?experiment)
ORDER BY ASC(?value)
LIMIT 5
```

Listing B.3: SPARQL Query Q3, extraction the parameters of one specific experiment. **moody:Experiment_NSGAII_00001** is an example of the URI of an *Experiment*.

Result:

?experiment	?problem	?value
moody:Experiment_3049	moody:Problem_ZDT4	6.58993E-1
moody:Experiment_1731	moody:Problem_ZDT4	6.59747E-1
moody:Experiment_3338	moody:Problem_ZDT4	6.60115E-1
moody:Experiment_3541	moody:Problem_ZDT4	6.60986E-1
moody:Experiment_3171	moody:Problem_ZDT4	6.61028E-1

Table B.3: Results of SPARQL Query Q3, extraction the parameters of one specific experiment.

```
PREFIX moody: <https://w3id.org/moody#>
PREFIX bigowl: <http://www.khaos.uma.es/lod/bigowl#>
SELECT DISTINCT ?problem_resolution ?experiment ?problem ?currentEvaluations ?hv_double
WHERE {
    VALUES ?algorithm { moody:Algorithm_NSGAII } .
    VALUES ?problem { moody:Problem_ZDT4 } .
    ?problem_resolution rdf:type moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?experiment moody:algorithmUsed ?algorithm .
    ?problem_resolution moody:currentNumberOfEvaluations ?currentEvaluations .
    ?problem_resolution moody:indicatorValue ?hvValue .
    ?hvValue moody:valueOfIndicator moody:QualityIndicator_HyperVolume .
    ?hvValue moody:hyperVolumeValue ?hv_double .
    ?experiment moody:parameterValue ?parameter_value_AR .
    ?parameter_value_AR moody:valueOfParameter moody:Parameter_AlgorithmResult .
    ?parameter_value_AR moody:algorithmResultValue ?value_AR .
    FILTER ( ?value_AR = "population" )
    ?experiment moody:parameterValue ?parameter_value_OPS .
    ?parameter_value_OPS moody:valueOfParameter moody:Parameter_OffspringPopulationSize .
    ?parameter_value_OPS moody:offspringPopulationSizeValue ?value_OPS .
    FILTER ( ?value_OPS = 100)
    ?experiment moody:parameterValue ?parameter_value_S .
    ?parameter_value_S moody:valueOfParameter moody:Parameter_Selection .
    ?parameter_value_S moody:selectionValue ?value_S .
    FILTER ( ?value_S = "tournament" )
    ?experiment moody:parameterValue ?parameter_value_STS .
    ?parameter_value_STS moody:valueOfParameter moody:Parameter_SelectionTournamentSize .
    ?parameter_value_STS moody:selectionTournamentSizeValue ?value_STS .
    FILTER ( ?value_STS = 2 )
    ?experiment moody:parameterValue ?parameter_value_C .
    ?parameter_value_C moody:valueOfParameter moody:Parameter_Crossover .
    ?parameter_value_C moody:crossoverValue ?value_C .
    FILTER ( ?value_C = "SBX" )
    ?experiment moody:parameterValue ?parameter_value_SCDI .
    ?parameter_value_SCDI moody:valueOfParameter moody:Parameter_SbxCrossoverDistributionIndex .
    ?parameter_value_SCDI moody:sbxCrossoverDistributionIndexValue ?value_SCDI .
    FILTER ( abs(?value_SCDI - 20.0) < 2.0)
    ?experiment moody:parameterValue ?parameter_value_M .
    ?parameter_value_M moody:valueOfParameter moody:Parameter_Mutation .
    ?parameter_value_M moody:mutationValue ?value_M .
    FILTER ( ?value_M = "polynomial" )
    ?experiment moody:parameterValue ?parameter_value_MP .
    ?parameter_value_MP moody:valueOfParameter moody:Parameter_MutationProbability .
    ?parameter_value_MP moody:mutationProbabilityValue ?value_MP .
    FILTER ( abs(?value_MP - 0.1) < 0.01)
    ?experiment moody:parameterValue ?parameter_value_PMDI .
    ?parameter_value_PMDI moody:valueOfParameter
        moody:Parameter_PolynomialMutationDistributionIndex .
    ?parameter_value_PMDI moody:polynomialMutationDistributionIndexValue ?value_PMDI .
    FILTER ( abs(?value_PMDI - 20.0) < 2.0)
}
ORDER BY ASC(?experiment)
```

Listing B.4: SPARQL Query Q4, search for algorithmic configurations that are compatible with pagmo.

```
BIND (
    ABS(1 - IF(?distanceXAverage2 < ?distanceXAverage,
        ?distanceXAverage2/?distanceXAverage,
        ?distanceXAverage/?distanceXAverage2)) +
    ABS(1 - IF(?distanceFAverage2 < ?distanceFAverage,
        ?distanceFAverage2/?distanceFAverage,
        ?distanceFAverage/?distanceFAverage2)) +
    ABS(1 - IF(?distanceXMaximum2 < ?distanceXMaximum,
        ?distanceXMaximum2/?distanceXMaximum,
        ?distanceXMaximum/?distanceXMaximum2)) +
    ABS(1 - IF(?distanceFMaximum2 < ?distanceFMaximum,
        ?distanceFMaximum2/?distanceFMaximum,
        ?distanceFMaximum/?distanceFMaximum2)) +
    ABS(1 - IF(?distanceXNonDomedMaximum2 < ?distanceXNonDomedMaximum,
        ?distanceXNonDomedMaximum2/?distanceXNonDomedMaximum,
        ?distanceXNonDomedMaximum/?distanceXNonDomedMaximum2)) +
    ABS(1 - IF(?distanceXNonDomedAverage2 < ?distanceXNonDomedAverage,
        ?distanceXNonDomedAverage2/?distanceXNonDomedAverage,
        ?distanceXNonDomedAverage/?distanceXNonDomedAverage2)) +
    ABS(1 - IF(?neighbourDistanceXMaximum2 < ?neighbourDistanceXMaximum,
        ?neighbourDistanceXMaximum2/?neighbourDistanceXMaximum,
        ?neighbourDistanceXMaximum/?neighbourDistanceXMaximum2)) +
    ABS(1 - IF(?neighbourDistanceXAverage2 < ?neighbourDistanceXAverage,
        ?neighbourDistanceXAverage2/?neighbourDistanceXAverage,
        ?neighbourDistanceXAverage/?neighbourDistanceXAverage2)) +
    ABS(1 - IF(?neighbourDistanceFMaximum2 < ?neighbourDistanceFMaximum,
        ?neighbourDistanceFMaximum2/?neighbourDistanceFMaximum,
        ?neighbourDistanceFMaximum/?neighbourDistanceFMaximum2)) +
    ABS(1 - IF(?neighbourDistanceFAverage2 < ?neighbourDistanceFAverage,
        ?neighbourDistanceFAverage2/?neighbourDistanceFAverage,
        ?neighbourDistanceFAverage/?neighbourDistanceFAverage2)) +
    ABS(1 - IF(?averagePropOfDomingNeigh2 < ?averagePropOfDomingNeigh,
        ?averagePropOfDomingNeigh2/?averagePropOfDomingNeigh,
        ?averagePropOfDomingNeigh/?averagePropOfDomingNeigh2)) +
    ABS(1 - IF(?averagePropOfDomedNeigh2 < ?averagePropOfDomedNeigh,
        ?averagePropOfDomedNeigh2/?averagePropOfDomedNeigh,
        ?averagePropOfDomedNeigh/?averagePropOfDomedNeigh2)) +
    ABS(1 - IF(?averagePropOfIncNeigh2 < ?averagePropOfIncNeigh,
        ?averagePropOfIncNeigh2/?averagePropOfIncNeigh,
        ?averagePropOfIncNeigh/?averagePropOfIncNeigh2)) +
    ABS(1 - IF(?averagePropOfLocNonDomedNeigh2 < ?averagePropOfLocNonDomedNeigh,
        ?averagePropOfLocNonDomedNeigh2/?averagePropOfLocNonDomedNeigh,
        ?averagePropOfLocNonDomedNeigh/?averagePropOfLocNonDomedNeigh2)) +
    ABS(1 - IF(?averagePropOfSupLocNonDomedNeigh2 < ?averagePropOfSupLocNonDomedNeigh,
        ?averagePropOfSupLocNonDomedNeigh2/?averagePropOfSupLocNonDomedNeigh,
        ?averagePropOfSupLocNonDomedNeigh/?averagePropOfSupLocNonDomedNeigh2)) +
    ABS(1 - IF(?neighCorOfAverageDistanceX2 < ?neighCorOfAverageDistanceX,
        ?neighCorOfAverageDistanceX2/?neighCorOfAverageDistanceX,
        ?neighCorOfAverageDistanceX/?neighCorOfAverageDistanceX2)) +
    ABS(1 - IF(?neighCorOfAverageDistanceF2 < ?neighCorOfAverageDistanceF,
        ?neighCorOfAverageDistanceF2/?neighCorOfAverageDistanceF,
        ?neighCorOfAverageDistanceF/?neighCorOfAverageDistanceF2)) +
    ABS(1 - IF(?proportionOfNonDomed2 < ?proportionOfNonDomed,
        ?proportionOfNonDomed2/?proportionOfNonDomed,
        ?proportionOfNonDomed/?proportionOfNonDomed2)) +
    ABS(1 - IF(?rankAverage2 < ?rankAverage,
        ?rankAverage2/?rankAverage,
        ?rankAverage/?rankAverage2)) +
    ABS(1 - IF(?rankMaximum2 < ?rankMaximum,
        ?rankMaximum2/?rankMaximum,
        ?rankMaximum/?rankMaximum2)) +
    ABS(1 - IF(?rankEntropy2 < ?rankEntropy,
        ?rankEntropy2/?rankEntropy,
        ?rankEntropy/?rankEntropy2))
as ?distance) .
```

Listing B.5: SPARQL implementation of the similarity metric between multi-objective problems based on their landscape characteristics defined in Equation 4.1.

```
PREFIX moody: <https://w3id.org/moody#>
PREFIX bigowl: <https://w3id.org/BIGOWLProblems/>
SELECT DISTINCT ?problem ?problem2 ?distance
WHERE {
    ?problem rdf:type bigowl:Problem .
    ?problem moody:numberOfObjectives ?numberOfObjectives .
    ?problem moody:distanceXAverage ?distanceXAverage .
    ?problem moody:distanceFAverage ?distanceFAverage .
    ?problem moody:distanceXMaximum ?distanceXMaximum .
    ?problem moody:distanceFMaximum ?distanceFMaximum .
    ?problem moody:distanceXNonDominatedMaximum ?distanceXNonDomedMaximum .
    ?problem moody:distanceXNonDominatedAverage ?distanceXNonDomedAverage .
    ?problem moody:neighbourDistanceXMaximum ?neighbourDistanceXMaximum .
    ?problem moody:neighbourDistanceXAverage ?neighbourDistanceXAverage .
    ?problem moody:neighbourDistanceFMaximum ?neighbourDistanceFMaximum .
    ?problem moody:neighbourDistanceFAverage ?neighbourDistanceFAverage .
    ?problem moody:averageProportionOfDominatingNeighbours ?averagePropOfDomingNeigh .
    ?problem moody:averageProportionOfDominatedNeighbours ?averagePropOfDomedNeigh .
    ?problem moody:averageProportionOfIncomparableNeighbours ?averagePropOfIncNeigh .
    ?problem moody:averageProportionOfLocallyNonDominatedNeighbours
        ?averagePropOfLocNonDomedNeigh .
    ?problem moody:averageProportionOfSuportedLocallyNonDominatedNeighbours
        ?averagePropOfSupLocNonDomedNeigh .
    ?problem moody:neighboursCorrelationOfAverageDistanceX ?neighCorOfAverageDistanceX .
    ?problem moody:neighboursCorrelationOfAverageDistanceF ?neighCorOfAverageDistanceF .
    ?problem moody:proportionOfNonDominated ?proportionOfNonDomed .
    ?problem moody:rankAverage ?rankAverage .
    ?problem moody:rankMaximum ?rankMaximum .
    ?problem moody:rankEntropy ?rankEntropy .

    ?problem2 rdf:type bigowl:Problem .
    ?problem2 moody:numberOfObjectives ?numberOfObjectives2 .
    ?problem2 moody:distanceXAverage ?distanceXAverage2 .
    ?problem2 moody:distanceFAverage ?distanceFAverage2 .
    ?problem2 moody:distanceXMaximum ?distanceXMaximum2 .
    ?problem2 moody:distanceFMaximum ?distanceFMaximum2 .
    ?problem2 moody:distanceXNonDominatedMaximum ?distanceXNonDomedMaximum2 .
    ?problem2 moody:distanceXNonDominatedAverage ?distanceXNonDomedAverage2 .
    ?problem2 moody:neighbourDistanceXMaximum ?neighbourDistanceXMaximum2 .
    ?problem2 moody:neighbourDistanceXAverage ?neighbourDistanceXAverage2 .
    ?problem2 moody:neighbourDistanceFMaximum ?neighbourDistanceFMaximum2 .
    ?problem2 moody:neighbourDistanceFAverage ?neighbourDistanceFAverage2 .
    ?problem2 moody:averageProportionOfDominatingNeighbours
        ?averagePropOfDomingNeigh2 .
    ?problem2 moody:averageProportionOfDominatedNeighbours ?averagePropOfDomedNeigh2 .
    ?problem2 moody:averageProportionOfIncomparableNeighbours ?averagePropOfIncNeigh2 .
    ?problem2 moody:averageProportionOfLocallyNonDominatedNeighbours
        ?averagePropOfLocNonDomedNeigh2 .
    ?problem2 moody:averageProportionOfSuportedLocallyNonDominatedNeighbours
        ?averagePropOfSupLocNonDomedNeigh2 .
    ?problem2 moody:neighboursCorrelationOfAverageDistanceX
        ?neighCorOfAverageDistanceX2 .
    ?problem2 moody:neighboursCorrelationOfAverageDistanceF
        ?neighCorOfAverageDistanceF2 .
    ?problem2 moody:proportionOfNonDominated ?proportionOfNonDomed2 .
    ?problem2 moody:rankAverage ?rankAverage2 .
    ?problem2 moody:rankMaximum ?rankMaximum2 .
    ?problem2 moody:rankEntropy ?rankEntropy2 .

    FILTER (?distance != "NaN"^^xsd:double)

    # Insert the BIND statement from Listing B.5
    BIND (... as ?distance) .
}
ORDER BY ASC(?distance)
```

Listing B.6: SPARQL Query Q5, implementation of the proposed similitude metric based on landscape characteristics. Requires inserting the BIND statement from Listing B.5.

Bibliography

[1] Naseer Ahmad, Shahid Kamal, Zulfiqar Ali Raza, and Tanveer Hussain. "Multi-objective optimization in the development of oil and water repellent cellulose fabric based on response surface methodology and the desirability function". In: *Materials Research Express* 4.3 (Mar. 2017), p. 035302. DOI: 10.1088/2053-1591/aa5f6a.

[2] José F. Aldana-Martín, Juan J. Durillo, and Antonio J. Nebro. "Evolver: Meta-optimizing multi-objective metaheuristics". In: *SoftwareX* 24 (2023), p. 101551. ISSN: 2352-7110. DOI: 10.1016/j.softx.2023.101551.

[3] José F. Aldana-Martín, María del Mar Roldán-García, Antonio J. Nebro, and José F. Aldana-Montes. "MOODY: An ontology-driven framework for standardizing multi-objective evolutionary algorithms". In: *Information Sciences* 661 (2024), p. 120184. ISSN: 0020-0255. DOI: 10.1016/j.ins.2024.120184.

[4] José F. Aldana-Martín, Antonio J. Nebro, Juan J. Durillo, and María del Mar Roldán García. "A Study About Meta-Optimizing the NSGA-II Multi-Objective Evolutionary Algorithm". In: *9th International Conference on Metaheuristics and Nature Inspired Computing (In press)*. Cham: Springer International Publishing, 2024.

[5] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. USA: Cambridge University Press, 2003. ISBN: 0521781760.

[6] Cristóbal Barba-González, José García-Nieto, María del Mar Roldán-García, Ismael Navas-Delgado, Antonio J. Nebro, and José F. Aldana-Montes. "BIGOWL: Knowledge centered Big Data analytics". In: *Expert Systems with Applications* 115 (2019), pp. 543–556. ISSN: 0957-4174.

[7] Cristóbal Barba-González, Antonio J. Nebro, José García-Nieto, María del Mar Roldán-García, Ismael Navas-Delgado, and José F. Aldana-Montes. "Injecting domain knowledge in multi-objective optimization problems:

A semantic approach". In: *Computer Standards & Interfaces* 78 (2021), p. 103546. ISSN: 0920-5489.

[8] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. "Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis". In: *Journal of Machine Learning Research* 18.77 (2017), pp. 1–36. URL: http://jmlr.org/papers/v18/16-305.html.

[9] Antonio Benítez-Hidalgo, Antonio J. Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. "jMetalPy: A Python framework for multi-objective optimization with metaheuristics". In: *Swarm and Evolutionary Computation* 51 (2019), p. 100598. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2019.100598.

[10] Nicola Beume, Boris Naujoks, and Michael Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume". In: *European Journal of Operational Research* 181.3 (2007), pp. 1653–1669. ISSN: 0377-2217.

[11] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. "Automatic Component-Wise Design of Multiobjective Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 20.3 (2016), pp. 403–417. DOI: 10.1109/TEVC.2015.2474158.

[12] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. "Automatic Configuration of Multi-objective Optimizers and Multi-objective Configuration". In: *High-Performance Simulation-Based Optimization*. Ed. by Thomas Bartz-Beielstein, Bogdan Filipič, Peter Korošec, and El-Ghazali Talbi. Cham: Springer International Publishing, 2020, pp. 69–92. ISBN: 978-3-030-18764-4.

[13] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. "Automatically Designing State-of-the-Art Multi- and Many-Objective Evolutionary Algorithms". In: *Evolutionary Computation* 28.2 (June 2020), pp. 195–226.

[14] Francesco Biscani and Dario Izzo. "A parallel global multiobjective framework for optimization: pagmo". In: *Journal of Open Source Software* 5.53 (2020), p. 2338. DOI: 10.21105/joss.02338.

[15] J. Blank and K. Deb. "pymoo: Multi-Objective Optimization in Python". In: *IEEE Access* 8 (2020), pp. 89497–89509.

[16] Aymeric Blot, Holger H. Hoos, Laetitia Jourdan, Marie-Éléonore Kessaci-Marmion, and Heike Trautmann. "MO-ParamILS: A Multi-objective Automatic Algorithm Configuration Framework". In: *Learning and Intelligent Optimization*. Ed. by Paola Festa, Meinolf Sellmann, and Joaquin Vanschoren. Cham: Springer International Publishing, 2016, pp. 32–47. ISBN: 978-3-319-50349-3.

[17] Christian Blum and Andrea Roli. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". In: *ACM Comput. Surv.* 35.3 (Sept. 2003), pp. 268–308. ISSN: 0360-0300. DOI: 10.1145/937503.937505.

[18] Ryan Boldi and Lee Spector. "Can the Problem-Solving Benefits of Quality Diversity Be Obtained without Explicit Diversity Maintenance?" In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. GECCO '23 Companion. Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 2152–2156. DOI: 10.1145/3583133.3596336.

[19] Jakob Bossek and Markus Wagner. "Generating instances with performance differences for more than just two algorithms". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '21. Lille, France: Association for Computing Machinery, 2021, pp. 1423–1432. ISBN: 9781450383516. DOI: 10.1145/3449726.3463165.

[20] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, and Tea Tušar. "Using Well-Understood Single-Objective Functions in Multiobjective Black-Box Optimization Test Suites". In: *Evolutionary Computation* 30.2 (June 2022), pp. 165–193. ISSN: 1063-6560. DOI: 10.1162/evco_a_00298.

[21] Dimo Brockhoff, Tea Tusar, Dejan Tusar, Tobias Wagner, Nikolaus Hansen, and Anne Auger. "Biobjective Performance Assessment with the COCO Platform". In: *CoRR* abs/1605.01746 (2016). arXiv: 1605.01746. URL: http://arxiv.org/abs/1605.01746.

[22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[23] Christian Leonardo Camacho-Villalón, Marco Dorigo, and Thomas Stützle. "The intelligent water drops algorithm: why it cannot be considered a novel algorithm". In: *Swarm Intelligence* 13.3 (Dec. 2019), pp. 173–192. ISSN: 1935-3820. DOI: 10.1007/s11721-019-00165-y.

[24] Luca Caneparo. "Semantic knowledge in generation of 3D layouts for decision-making". In: *Automation in Construction* 134 (2022), p. 104012. ISSN: 0926-5805.

[25] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. "Quality-Diversity Optimization: A Novel Branch

of Stochastic Optimization". In: *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. Ed. by Panos M. Pardalos, Varvara Rasskazova, and Michael N. Vrahatis. Cham: Springer International Publishing, 2021, pp. 109–135. ISBN: 978-3-030-66515-9. DOI: 10.1007/978-3-030-66515-9_4.

[26] Yen-Sheng Chen. "Multiobjective optimization of complex antenna structures using response surface models". In: *International Journal of RF and Microwave Computer-Aided Engineering* 26.1 (2016), pp. 62–71. DOI: 10.1002/mmce.20939.

[27] Xianghua Chu, Jiayun Wang, Shuxiang Li, Linya Huang, Qiu He, Guodan Bao, and Wei Zhao. "Meta-feature Extraction for Multi-objective Optimization Problems". In: *Neural Computing for Advanced Applications*. Ed. by Haijun Zhang, Zhi Yang, Zhao Zhang, Zhou Wu, and Tianyong Hao. Singapore: Springer Singapore, 2021, pp. 432–445. ISBN: 978-981-16-5188-5.

[28] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Second. ISBN 978-0-387-33254-3. New York: Springer, Sept. 2007.

[29] Carlos A. Coello Coello and Margarita Reyes Sierra. "A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm". In: *MICAI 2004: Advances in Artificial Intelligence*. Ed. by Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 688–697. ISBN: 978-3-540-24694-7.

[30] Raphaël Cosson, Bilel Derbel, Arnaud Liefooghe, Sébastien Verel, Hernan Aguirre, Qingfu Zhang, and Kiyoshi Tanaka. "Cost-vs-accuracy of sampling in multi-objective combinatorial exploratory landscape analysis". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 493–501. ISBN: 9781450392372. DOI: 10.1145/3512290.3528731.

[31] M. Dean and G. Schreiber. *OWL Web Ontology Language Reference*. Tech. rep. W3C Recommendation, 10 February 2004, 2004. URL: http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[32] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, 2001.

[33] Kalyanmoy Deb. "Multi-objective optimization using evolutionary algorithms". In: *Wiley-Interscience series in systems and optimization*. 2001.

[34] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. "A fast elitist non-dominated sorting genetic algorithm for multi-objective op-

timization: NSGA-II". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1917 (2000), pp. 849–858.

[35] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[36] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. "Scalable Test Problems for Evolutionary Multiobjective Optimization". In: *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. Ed. by Ajith Abraham, Lakhmi Jain, and Robert Goldberg. London: Springer London, 2005, pp. 105–145. ISBN: 978-1-84628-137-2.

[37] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N. Suganthan, Carlos A. Coello Coello, and Francisco Herrera. "Bio-inspired computation: Where we stand and what's next". In: *Swarm and Evolutionary Computation* 48 (2019), pp. 220–250. ISSN: 2210-6502. DOI: 10.1016/j.swevo.2019.04.008.

[38] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (Dec. 2006), pp. 1–30. ISSN: 1532-4435.

[39] Konstantin Dietrich, Diederick Vermetten, Carola Doerr, and Pascal Kerschke. *Impact of Training Instance Selection on Automated Algorithm Selection Models for Numerical Black-box Optimization*. 2024. arXiv: 2404.07539 [cs.NE].

[40] Daniel Doblas, Antonio J. Nebro, Manuel López-Ibáñez, José García-Nieto, and Carlos A. Coello Coello. "Automatic Design of Multi-objective Particle Swarm Optimizers". In: *Swarm Intelligence*. Ed. by Marco Dorigo, Heiko Hamann, Manuel López-Ibáñez, José García-Nieto, Andries Engelbrecht, Carlo Pinciroli, Volker Strobel, and Christian Camacho-Villalón. Cham: Springer International Publishing, 2022, pp. 28–40. ISBN: 978-3-031-20176-9.

[41] Juan J. Durillo and Antonio J. Nebro. "jMetal: A Java framework for multi-objective optimization". In: *Advances in Engineering Software* 42.10 (2011), pp. 760–771. ISSN: 0965-9978.

[42] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. "A study of master-slave approaches to parallelize NSGA-II". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. 2008, pp. 1–8.

[43] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. "On the Effect of the Steady-State Selection Scheme in Multi-Objective Genetic Algorithms". In: *Evolutionary Multi-Criterion Optimization*. Ed. by Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 183–197. ISBN: 978-3-642-01020-0.

[44] Lisa Ehrlinger and Wolfram Wöß. "Towards a definition of knowledge graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016), p. 2.

[45] Michael T. M. Emmerich and André H. Deutz. "A tutorial on multiobjective optimization: fundamentals and evolutionary methods". In: *Natural Computing* 17.3 (Sept. 2018), pp. 585–609. ISSN: 1572-9796.

[46] Absalom E. Ezugwu, Amit K. Shukla, Rahul Nath, Andronicus A. Akinyelu, Jeffery O. Agushaka, Haruna Chiroma, and Pranab K. Muhuri. "Metaheuristics: a comprehensive overview and classification along with bibliometric analysis". In: *Artificial Intelligence Review* 54.6 (Aug. 2021), pp. 4237–4316. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09952-0.

[47] A. Farhang-Mehr and S. Azarm. "Entropy-based multi-objective genetic algorithm for design optimization". In: *Structural and Multidisciplinary Optimization* 24.5 (2002), pp. 351–361. DOI: 10.1007/s00158-002-0247-6.

[48] Matthew Fontaine and Stefanos Nikolaidis. "Covariance Matrix Adaptation MAP-Annealing". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 456–465. DOI: 10.1145/3583131.3590389. URL: https://doi.org/10.1145/3583131.3590389.

[49] Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. "Covariance matrix adaptation for the rapid illumination of behavior space". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO '20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 94–102. DOI: 10.1145/3377930.3390232. URL: https://doi.org/10.1145/3377930.3390232.

[50] Alberto Franzin and Thomas Stützle. "A Causal Framework for Understanding Optimisation Algorithms". In: *Trustworthy AI - Integrating Learning, Optimization and Reasoning*. Ed. by Fredrik Heintz, Michela Milano, and Barry O'Sullivan. Cham: Springer International Publishing, 2021, pp. 140–145. ISBN: 978-3-030-73959-1.

[51] Alberto Franzin and Thomas Stützle. *A causal framework for optimization algorithms*. Tech. rep. TR/IRIDIA/2022-007. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2022.

[52] T. Ganesan, I. Elamvazuthi, Ku Zilati Ku Shaari, and P. Vasant. "Swarm intelligence and gravitational search algorithm for multi-objective optimization of synthesis gas production". In: *Applied Energy* 103 (2013), pp. 368–374. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2012.09.059.

[53] Long Gao, Gegentana, Zhongze Liu, Baizhong Sun, Deyong Che, and Shaohua Li. "Multi-objective optimization of thermal performance of packed bed latent heat thermal storage system based on response surface method". In: *Renewable Energy* 153 (2020), pp. 669–680. ISSN: 0960-1481. DOI: 10.1016/j.renene.2020.01.157.

[54] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. *A Survey of Quantization Methods for Efficient Neural Network Inference*. 2021. arXiv: 2103.13630 [cs.CV].

[55] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media, 2006.

[56] Tushar Goel, Rajkumar Vaidyanathan, Raphael T. Haftka, Wei Shyy, Nestor V. Queipo, and Kevin Tucker. "Response surface approximation of Pareto optimal front in multi-objective optimization". In: *Computer Methods in Applied Mechanics and Engineering* 196.4 (2007), pp. 879–893. ISSN: 0045-7825. DOI: 10.1016/j.cma.2006.07.010.

[57] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201157675.

[58] T. R. Gruber. "A translation approach to portable ontologies". In: *Knowledge Acquisition,* 5.2 (1993), pp. 199–220.

[59] Fangqing Gu, Hai-Lin Liu, and Kay Chen Tan. "A multiobjective evolutionary algorithm using dynamic weight design method". In: *International Journal of Innovative Computing, Information and Control* 8.5 B (2012), pp. 3677–3688.

[60] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. "CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 639–647. DOI: 10.1145/3512290.3528827.

[61] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. "COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting". In: *Optimization Methods and Software* 36 (1 2021), pp. 114–144. DOI: https://doi.org/10.1080/10556788.2020.1808977.

[62] N. Hansen and A. Ostermeier. “Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation”. In: *Proceedings of IEEE International Conference on Evolutionary Computation*. 1996, pp. 312–317. DOI: 10.1109/ICEC.1996.542381.

[63] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, Dejan Tusar, and Tea Tusar. “COCO: Performance Assessment”. In: *CoRR* abs/1605.03560 (2016). arXiv: 1605.03560. URL: http://arxiv.org/abs/1605.03560.

[64] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. Tech. rep. Mar. 2013. URL: https://www.w3.org/TR/sparql11-query/.

[65] Linjun He, Hisao Ishibuchi, Anupam Trivedi, Handing Wang, Yang Nan, and Dipti Srinivasan. “A Survey of Normalization Methods in Multiobjective Evolutionary Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 25.6 (2021), pp. 1028–1048. DOI: 10.1109/TEVC.2021.3076514.

[66] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. *Deep Learning Scaling is Predictable, Empirically*. 2017. arXiv: 1712.00409 [cs.LG].

[67] Pascal Hitzler. “A Review of the Semantic Web Field”. In: *Commun. ACM* 64.2 (Jan. 2021), pp. 76–83. ISSN: 0001-0782. DOI: 10.1145/3397512.

[68] Pascal Hitzler, Jens Lehmann, and Axel Polleres. “Logics for the Semantic Web”. In: *Computational Logic*. Ed. by Jörg H. Siekmann. Vol. 9. Handbook of the History of Logic. North-Holland, 2014, pp. 679–710.

[69] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL].

[70] Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, and Dmitry Tsarkov. “OWL rules: A proposal and prototype implementation”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.1 (2005), pp. 23–40.

[71] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. “Practical Reasoning for Very Expressive Description Logics”. In: *Log. J. IGPL* 8 (2000), pp. 239–263.

[72] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[73] Changwu Huang, Yuanxiang Li, and Xin Yao. "A survey of automatic parameter tuning methods for metaheuristics". In: *IEEE Transactions on Evolutionary Computation* 24.2 (2019), pp. 201–216.

[74] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. 2023. arXiv: 2311.05232 [cs.CL].

[75] S. Huband, P. Hingston, L. Barone, and Lyndon While. "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit". In: *IEEE Transactions on Evolutionary Computation* 10.5 (Oct. 2006), pp. 477–506.

[76] Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. "A Scalable Multi-objective Test Problem Toolkit". In: *Evolutionary Multi-Criterion Optimization*. Ed. by Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 280–295. ISBN: 978-3-540-31880-4.

[77] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. "ParamILS: An Automatic Algorithm Configuration Framework". In: *J. Artif. Int. Res.* 36.1 (Sept. 2009), pp. 267–306. ISSN: 1076-9757.

[78] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. "Modified Distance Calculation in Generational Distance and Inverted Generational Distance". In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Cham: Springer International Publishing, 2015, pp. 110–125. ISBN: 978-3-319-15892-1.

[79] Hisao Ishibuchi, Lie Meng Pang, and Ke Shang. "A New Framework of Evolutionary Multi-Objective Algorithms with an Unbounded External Archive". In: *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*. Ed. by Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, 2020, pp. 283–290.

[80] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang,

Timothée Lacroix, and William El Sayed. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].

[81] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. *Mixtral of Experts*. 2024. arXiv: 2401.04088 [cs.LG].

[82] Gopal K Kanji. *100 statistical tests*. Sage, 2006.

[83] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].

[84] C. Maria Keet, Agnieszka Ławrynowicz, Claudia d'Amato, Alexandros Kalousis, Phong Nguyen, Raul Palma, Robert Stevens, and Melanie Hilario. "The Data Mining OPtimization Ontology". In: *Journal of Web Semantics* 32 (2015), pp. 43–53. ISSN: 1570-8268.

[85] Joshua D. Knowles, Lothar Thiele, and Eckart Zitzler. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. en. Report. Zurich, 2006.

[86] Ana Kostovska, Diederick Vermetten, Carola Doerr, Sašo Džeroski, Panče Panov, and Tome Eftimov. "OPTION: Optimization Algorithm Benchmarking Ontology". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '21. Lille, France: Association for Computing Machinery, 2021, pp. 239–240. ISBN: 9781450383516.

[87] Ludmila I. Kuncheva and Christopher J. Whitaker. "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy". In: *Machine Learning* 51.2 (May 2003), pp. 181–207. ISSN: 1573-0565. DOI: 10.1023/A:1022859003006. URL: https://doi.org/10.1023/A:1022859003006.

[88] H. Lebesgue. "Intégrale, Longueur, Aire". In: *Annali di Matematica Pura ed Applicata (1898-1922)* 7.1 (Dec. 1902), pp. 231–359. ISSN: 0373-3114. DOI: 10.1007/BF02420592.

[89] Hui Li and Qingfu Zhang. "Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II". In: *Trans. Evol. Comp* 13.2 (Apr. 2009), pp. 284–302. ISSN: 1089-778X.

[90] Longmei Li, Iryna Yevseyeva, Vitor Basto-Fernandes, Heike Trautmann, Ning Jing, and Michael Emmerich. *An Ontology of Preference-Based Multiobjective Metaheuristics*. 2017. arXiv: 1609.08082 [cs.NE].

[91] Longmei Li, Iryna Yevseyeva, Vitor Basto-Fernandes, Heike Trautmann, Ning Jing, and Michael Emmerich. "Building and Using an Ontology of Preference-Based Multiobjective Evolutionary Algorithms". In: *Evolutionary Multi-Criterion Optimization*. Ed. by Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme. Cham: Springer International Publishing, 2017, pp. 406–421. ISBN: 978-3-319-54157-0.

[92] Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. "Multiobjective optimization for crash safety design of vehicles using stepwise regression model". In: *Structural and Multidisciplinary Optimization* 35.6 (June 2008), pp. 561–569. ISSN: 1615-1488. DOI: 10.1007/s00158-007-0163-x.

[93] Arnaud Liefooghe. "Landscape analysis and heuristic search for multiobjective optimization". Habilitation à diriger des recherches. Université de Lille, June 2022. URL: https://hal.science/tel-03758552.

[94] Arnaud Liefooghe, Sébastien Verel, Benjamin Lacroix, Alexandru-Ciprian Zăvoianu, and John McCall. "Landscape Features and Automated Algorithm Selection for Multi-Objective Interpolated Continuous Optimisation Problems". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '21. Lille, France: Association for Computing Machinery, 2021, pp. 421–429. ISBN: 9781450383509.

[95] Ricardo Henrique Remes de Lima and Aurora Trinidad Ramirez Pozo. "A study on auto-configuration of Multi-Objective Particle Swarm Optimization Algorithm". In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2017, pp. 718–725.

[96] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization". In: *Journal of Machine Learning Research* 23.54 (2022), pp. 1–9. URL: http://jmlr.org/papers/v23/21-0888.html.

[97] Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. *Large Language Model for Multi-objective Evolutionary Optimization*. 2024. arXiv: 2310.12541 [cs.NE].

[98] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. *An Example of Evolutionary Computation + Large Language Model Beating Human: Design of Efficient Guided Local Search*. 2024. arXiv: 2401.02051 [cs.NE].

[99] Jingfa Liu, Yi Dong, Zhaoxia Liu, and Duanbing Chen. "Applying ontology learning and multi-objective ant colony optimization method for focused crawling to meteorological disasters domain knowledge". In: *Expert Systems with Applications* 198 (2022).

[100] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. "Lost in the Middle: How Language Models Use Long Contexts". In: *Transactions of the Association for Computational Linguistics* 12 (Feb. 2024), pp. 157–173. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00638.

[101] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. *Large Language Models as Evolutionary Optimizers*. 2023. arXiv: 2310.19046 [cs.NE].

[102] Manuel López-Ibañez and Thomas Stützle. "The Automatic Design of Multiobjective Ant Colony Optimization Algorithms". In: *IEEE Transactions on Evolutionary Computation* 16.6 (2012), pp. 861–875. DOI: 10.1109/TEVC.2011.2182651.

[103] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58. ISSN: 2214-7160.

[104] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. *The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits*. 2024. arXiv: 2402.17764 [cs.CL].

[105] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. DOI: 10.1214/aoms/1177730491.

[106] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245. ISSN: 00401706. URL: http://www.jstor.org/stable/1268522 (visited on 01/23/2024).

[107] Dirk Merkel. "Docker: lightweight Linux containers for consistent development and deployment". In: *Linux Journal* 2014 (2014), p. 2.

[108] Mustafa Mısır and Michèle Sebag. "Alors: An algorithm recommender system". In: *Artificial Intelligence* 244 (2017). Combining Constraint Solving with Mining and Learning, pp. 291–314. ISSN: 0004-3702. DOI: 10.1016/j.artint.2016.12.001.

[109] Jean-Baptiste Mouret and Jeff Clune. *Illuminating search spaces by mapping elites*. 2015. arXiv: 1504.04909 [cs.AI].

[110] A.J. Nebro, J.J. Durilllo, J. García-Nieto, C.A. Coello Coello, F. Luna, and E. Alba. "SMPSO: a new PSO-Based metaheuristic for multi-objective optimization". In: *2009 IEEE Symposium on Computational Intelligence in Multi-criteria decision-making (MCDM 2009)*. IEEE Computer Society, 2009, pp. 66–73. ISBN: 978-1-4244-2764-2. DOI: 10.1109/MCDM.2009.4938830.

[111] Antonio J Nebro, Manuel López-Ibáñez, José García-Nieto, and Carlos A Coello Coello. "On the automatic design of multi-objective particle swarm optimizers: experimentation and analysis". In: *Swarm Intelligence* (2023), pp. 1–35.

[112] Antonio J. Nebro and Juan J. Durillo. "On the Effect of Applying a Steady-State Selection Scheme in the Multi-Objective Genetic Algorithm NSGA-II". In: *Nature-Inspired Algorithms for Optimisation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 435–456. ISBN: 978-3-642-00267-0. DOI: 10.1007/978-3-642-00267-0_16.

[113] Antonio J. Nebro, Juan J. Durillo, and Matthieu Vergne. "Redesigning the jMetal multi-objective optimization framework". In: *GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference* (July 2015), pp. 1093–1100.

[114] Antonio J. Nebro, Jesús Galeano-Brajones, Francisco Luna, and Carlos A. Coello Coello. "Is NSGA-II Ready for Large-Scale Multi-Objective Optimization?" In: *Mathematical and Computational Applications* 27.6 (2022). ISSN: 2297-8747. DOI: 10.3390/mca27060103.

[115] Antonio J. Nebro, Manuel López-Ibáñez, Cristóbal Barba-González, and José García-Nieto. "Automatic Configuration of NSGA-II with JMetal and Irace". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. Prague, Czech Republic: Association for Computing Machinery, 2019, pp. 1374–1381. ISBN: 9781450367486.

[116] Antonio J. Nebro, Javier Pérez-Abad, José F. Aldana-Martin, and José García-Nieto. "Evolving a Multi-objective Optimization Framework". In: *Applied Optimization and Swarm Intelligence*. Ed. by Eneko Osaba and Xin-She Yang. Singapore: Springer Singapore, 2021, pp. 175–198. ISBN: 978-981-16-0662-5.

[117] Christoph Neumüller, Stefan Wagner, Gabriel Kronberger, and Michael Affenzeller. "Parameter meta-optimization of metaheuristic optimization algorithms". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6927 LNCS.PART 1 (2012). Cited by: 22, pp. 367–374.

[118] Natasha F. Noy and Deborah L. McGuinness. “Ontology Development 101: A Guide to Creating Your First Ontology”. In: *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880* (Mar. 2001).

[119] P. C. Padhi, S. S. Mahapatra, S. N. Yadav, and D. K. Tripathy. “Multi-Objective Optimization of Wire Electrical Discharge Machining (WEDM) Process Parameters Using Weighted Sum Genetic Algorithm Approach”. In: *Journal of Advanced Manufacturing Systems* 15.02 (2016), pp. 85–100. DOI: 10.1142/S0219686716500086. URL: https://doi.org/10.1142/S0219686716500086.

[120] Sinno Jialin Pan and Qiang Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010). DOI: 10.1109/TKDE.2009.191.

[121] Lie Meng Pang, Hisao Ishibuchi, and Ke Shang. “Offline Automatic Parameter Tuning of MOEA/D Using Genetic Algorithm”. In: *IEEE Symposium Series on Computational Intelligence, SSCI 2019, Xiamen, China, December 6-9, 2019*. IEEE, 2019, pp. 1889–1897.

[122] Lie Meng Pang, Hisao Ishibuchi, and Ke Shang. “Using a Genetic Algorithm-based Hyper-heuristic to Tune MOEA/D for a Set of Various Test Problems”. In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. 2021, pp. 1486–1494. DOI: 10.1109/CEC45853.2021.9504748.

[123] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. “Semantics and Complexity of SPARQL”. In: *ACM Trans. Database Syst.* 34.3 (Sept. 2009), 16:1–16:45. ISSN: 0362-5915.

[124] R. Polikar. “Ensemble based systems in decision making”. In: *IEEE Circuits and Systems Magazine* 6.3 (2006), pp. 21–45. DOI: 10.1109/MCAS.2006.1688199.

[125] Lorien Pratt and Barbara Jennings. “A Survey of Connectionist Network Reuse Through Transfer”. In: *Learning to Learn*. Ed. by Sebastian Thrun and Lorien Pratt. Boston, MA: Springer US, 1998, pp. 19–43. ISBN: 978-1-4615-5529-2. DOI: 10.1007/978-1-4615-5529-2_2.

[126] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. “Quality Diversity: A New Frontier for Evolutionary Computation”. In: *Frontiers in Robotics and AI* 3 (2016). ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00040.

[127] Jinda Qi, Lan Ding, and Samsung Lim. “A Decision-Making Framework to Support Urban Heat Mitigation by Local Governments”. In: *Resources, Conservation and Recycling* 184 (2022).

[128] Alec Radford and Karthik Narasimhan. “Improving Language Understanding by Generative Pre-Training”. In: 2018.

[129] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. “Language Models are Unsupervised Multitask Learners”. In: *OpenAI research* (2019).

[130] Priya Ranganathan. “An Introduction to Statistics: Choosing the Correct Statistical Test”. en. In: *Indian J Crit Care Med* 25.Suppl 2 (May 2021), S184–S186.

[131] Matthias C. Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A. Gould, and Uli Sauerland. “Risks and Benefits of Large Language Models for the Environment”. In: *Environmental Science & Technology* 57.9 (2023). PMID: 36821477, pp. 3464–3466. DOI: 10.1021/acs.est.3c01106.

[132] Matthew Rocklin. “Dask: Parallel Computation with Blocked algorithms and Task Scheduling”. In: *Proceedings of the 14th Python in Science Conference*. Ed. by Kathryn Huff and James Bergstra. 2015, pp. 130–136.

[133] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. *From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference*. 2023. arXiv: 2310.03003 [cs.CL].

[134] Salam Al-Sarayrah, Dareen Abulail, and Khaled Shaalan. “Understanding the Impact of the Ontology of Semantic Web in Knowledge Representation: A Systematic Review”. In: *Recent Innovations in Artificial Intelligence and Smart Applications*. Ed. by Mostafa Al-Emran and Khaled Shaalan. Cham: Springer International Publishing, 2022, pp. 277–299. ISBN: 978-3-031-14748-7.

[135] Guus Schreiber and Yves Raimond. *RDF 1.1 Primer*. Tech. rep. June 2014. URL: https://www.w3.org/TR/rdf11-primer/.

[136] S. S. SHAPIRO and M. B. WILK. “An analysis of variance test for normality (complete samples)†”. In: *Biometrika* 52.3-4 (Dec. 1965), pp. 591–611. ISSN: 0006-3444. DOI: 10.1093/biomet/52.3-4.591. eprint: https://academic.oup.com/biomet/article-pdf/52/3-4/591/962907/52-3-4-591.pdf.

[137] Robert DC Shearer, Boris Motik, and Ian Horrocks. “Hermit: A highly-efficient OWL reasoner.” In: *Owled*. Vol. 432. 2008, p. 91.

[138] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2020. arXiv: 1909.08053 [cs.CL].

[139] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A practical OWL-DL reasoner”. In: *Journal of Web Semantics* 5.2 (2007). Software Engineering and the Semantic Web, pp. 51–53. ISSN: 1570-8268.

[140] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzel, Suzanna Lewis, and The OBI Consortium. "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration". In: *Nature Biotechnology* 25.11 (Nov. 2007), pp. 1251–1255. ISSN: 1546-1696. DOI: 10.1038/nbt1346.

[141] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. *Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model*. 2022. arXiv: 2201.11990 [cs.CL].

[142] Peter Sollich and Anders Krogh. "Learning with ensembles: how overfitting can be useful". In: *Proceedings of the 8th International Conference on Neural Information Processing Systems*. NIPS'95. Denver, Colorado: MIT Press, 1995, pp. 190–196.

[143] Kenneth Sörensen. "Metaheuristics—the metaphor exposed". In: *International Transactions in Operational Research* 22.1 (2015), pp. 3–18. DOI: https://doi.org/10.1111/itor.12001.

[144] C. Spearman. "The Proof and Measurement of Association between Two Things". In: *The American Journal of Psychology* 15.1 (1904), pp. 72–101. ISSN: 00029556. DOI: 10.2307/F1412159. URL: http://www.jstor.org/stable/1412159 (visited on 02/02/2024).

[145] Steffen Staab and Rudi Studer. *Handbook on Ontologies*. Springer, 2010.

[146] Abdussamet Subasi, Bayram Sahin, and Irfan Kaymaz. "Multi-objective optimization of a honeycomb heat sink using Response Surface Method". In: *International Journal of Heat and Mass Transfer* 101 (2016), pp. 295–302. ISSN: 0017-9310. DOI: 10.1016/j.ijheatmasstransfer.2016.05.012.

[147] Ryoji Tanabe and Hisao Ishibuchi. "An easy-to-use real-world multi-objective optimization problem suite". In: *Applied Soft Computing* 89 (2020), p. 106078. ISSN: 1568-4946.

[148] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. "PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]". In: *IEEE Computational Intelligence Magazine* 12.4 (2017), pp. 73–87. DOI: 10.1109/MCI.2017.2742868.

[149] Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. "A Recommender System for Metaheuristic Algorithms for Continuous Optimization Based on Deep Recurrent Neural Networks". In: *IEEE Transactions on Artificial Intelligence* 1.1 (2020), pp. 5–18. DOI: 10.1109/TAI.2020.3022339.

[150] Bryon Tjanaka, Matthew C Fontaine, David H Lee, Yulun Zhang, Nivedit Reddy Balam, Nathaniel Dennler, Sujay S Garlanka, Nikitas Dimitri Klapsis, and Stefanos Nikolaidis. "pyribs: A Bare-Bones Python Library for Quality Diversity Optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 220–229. DOI: 10.1145/3583131.3590374. URL: https://doi.org/10.1145/3583131.3590374.

[151] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[152] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[153] Rajkumar Vaidyanathan, P. Kevin Tucker, Nilay Papila, and Wei Shyy. "Computational-Fluid-Dynamics-Based Design Optimization for Single-Element Rocket Injector". In: *Journal of Propulsion and Power* 20.4 (2004), pp. 705–717. DOI: 10.2514/1.11464. URL: https://doi.org/10.2514/1.11464.

[154] David A Van Veldhuizen, Gary B Lamont, et al. “Evolutionary computation and convergence to a pareto front”. In: *Late breaking papers at the genetic programming 1998 conference*. 1998, pp. 221–228.

[155] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[156] Sébastien Verel, Arnaud Liefooghe, Laetitia Jourdan, and Clarisse Dhaenens. “On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives”. In: *European Journal of Operational Research* 227.2 (2013), pp. 331–342. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.12.019.

[157] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C ’t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3.1 (Mar. 2016), p. 160018. ISSN: 2052-4463.

[158] Paul Witherell, Sundar Krishnamurty, and Ian R Grosse. “Ontologies for supporting engineering design optimization”. In: (2007).

[159] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[160] D.H. Wolpert and W.G. Macready. “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893.

[161] Chao Wu, Sannyuya Liu, Zeyu Zeng, Mao Chen, Adi Alhudhaif, Xiangyang Tang, Fayadh Alenezi, Norah Alnaim, and Xicheng Peng. “Knowledge graph-based multi-context-aware recommendation algorithm”. In: *Information Sciences* 595 (2022), pp. 179–194. ISSN: 0020-0255. DOI: 10.1016/j.ins.2022.02.054.

[162] Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. *Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap*. 2024. arXiv: 2401.10034 [cs.NE].

[163] Jin Xu, Fuwu Yan, Yan Li, Zhenchao Yang, and Long Li. “Multiobjective Optimization of Milling Parameters for Ultrahigh-Strength Steel AF1410 Based on the NSGA-II Method”. In: *Advances in Materials Science and Engineering* 2020 (July 2020), p. 8796738. ISSN: 1687-8434. DOI: 10.1155/2020/8796738.

[164] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. 2023. arXiv: 2312.12148 [cs.CL].

[165] Estefania Yap, Mario A. Muñoz, Kate Smith-Miles, and Arnaud Liefooghe. “Instance Space Analysis of Combinatorial Multi-objective Optimization Problems”. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. Glasgow, United Kingdom: IEEE Press, 2020, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185664.

[166] Wenjie Yi, Rong Qu, Licheng Jiao, and Ben Niu. “Automated Design of Metaheuristics Using Reinforcement Learning within a Novel General Search Framework”. In: *IEEE Transactions on Evolutionary Computation* (2022), pp. 1–1. DOI: 10.1109/TEVC.2022.3197298.

[167] Saúl Zapotecas-Martínez, Abel García-Nájera, and Adriana Menchaca-Méndez. “Engineering applications of multi-objective evolutionary algorithms: A test suite of box-constrained real-world problems”. In: *Engineering Applications of Artificial Intelligence* 123 (2023), p. 106192. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.106192.

[168] Kangkang Zhang and Yan Song. “A new multi-objective optimization algorithm based on combined swarm intelligence and Monte Carlo simulation”. In: *Information Sciences* 610 (2022), pp. 759–776. ISSN: 0020-0255. DOI: 10.1016/j.ins.2022.08.035.

[169] Qingfu Zhang and Hui Li. “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”. In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731.

[170] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. "Multiobjective optimization test instances for the CEC 2009 special session and competition". In: *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report* 264 (2008), pp. 1–30.

[171] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. *A Survey of Large Language Models*. 2023. arXiv: 2303.18223 [cs.CL].

[172] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. *A Comprehensive Survey on Transfer Learning*. 2020. arXiv: 1911.02685 [cs.LG].

[173] E. Zitzler and L. Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach". In: *IEEE Transactions on Evolutionary Computation* 3.4 (1999), pp. 257–271.

[174] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. "Performance assessment of multiobjective optimizers: an analysis and review". In: *IEEE Transactions on Evolutionary Computation* 7.2 (2003), pp. 117–132. DOI: 10.1109/TEVC.2003.810758.

[175] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results". In: *Evolutionary Computation* 8.2 (2000), pp. 173–195. DOI: 10.1162/106365600568202.

[176] Eckart Zitzler and Simon Künzli. "Indicator-Based Selection in Multiobjective Search". In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 832–842. ISBN: 978-3-540-30217-9.

[177] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm". In: *TIK-report* 103 (2001).

[178] Ying Zuo, Yuqi Wang, Yuanjun Laili, T. Warren Liao, and Fei Tao. "An evolutionary algorithm recommendation method with a case study in flow shop scheduling". In: *The International Journal of Advanced Manufacturing Technology* 109.3 (July 2020), pp. 781–796. ISSN: 1433-3015. DOI: 10.1007/s00170-020-05471-y.

Index